DRAGON USER

The independent Dragon magazine

February 1988

Contents

Brian Cadge is on holiday

Editorial

HAPPY NEW YEAR. I've been waiting for weeks to say that. To you, it may be late January, cold, wet and miserable, but to us it's the 4th of January, cold, wet, etc. Scot Press, most of British industry and the Post Office (nine days for Gordon Lee's copy to reach us, first class) stop for Christmas and the New Year, but not the Dragon.

John Penn Software has been in touch to announce a Dragon and Tandy show next April at Cardiff Wales Airport. The Penns are looking for demonstrators as well as retailers. See page 21 for further details. I was only mildly disappointed to discover that what I thought was Purple Car Painting was, in fact, Ample Car Parking, but perhaps Mr Makin can arrange something . . .

Requests for hardware projects are being met with a vengeance this month with a blockbusting experimental interface project. Although the article is long, the individual sections are in reality quite simple and described in detail, so all would-be hardware enthusiasts can have a go.

Many thanks from everybody at and around *Dragon User* to our subscribers and advertisers, who are the real reason the Dragon keeps going. Here's to another year . . .

Telephone number
(All departments)
437 4343

Editor
HELEN ARMSTRONG

Production Editor
DRAGON EDITORIAL

Administration
CAROL FRITH

Advertisement Manager
DRAGON EDITORIAL

Marketing Manager
HELEN PERRY

Managing Editor
PETER KANE

Publishing Director
JENNY IRELAND

Subscriptions
UK £14 for 12 issues
Overseas (surface) £20 for 12 issues
ISSN 0265-177 Telex 296275
Dragon User 12/13 Little Newport Street
London WC2H 7PP
US address c/o Business Press International
205 East 42nd St New York NY 10017
Published by Scot Press Ltd
© Scot Press 1986
Typesetting and Production by Artext Limited
London NW1
Printed by Headley Brothers Ltd Ashford Kent
Registered at the Post Office as a newspaper
Dragon and its logo are trademarks of
Eurohard Ltd

How to submit articles

The quality of the material we can publish in *Dragon User* each month will to a very great extent depend on the quality of the discoveries that you can make with your Dragon. The Dragon computer was launched on to the market with a powerful version of Basic, but with very poor documentation.

Articles which are submitted to *Dragon User* for publication should not be more than 3000 words long. All submissions should be typed. Please leave wide margins and a double space between each line. Programs should, whenever possible, be computer printed on plain white paper and be accompanied by a tape of the program.

We cannot guarantee to return every submitted article or program, so please keep a copy. If you want to have your program returned you must include a stamped addressed envelope.

Letters

This is your chance to air your views — send your tips, compliments and complaints to Letters Page, *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP

Not for killing

IN answer to Mike Hides's letter (December 1987) regarding the problem of quick disc filling, I have used the disc version of *Telewriter* since 1985 with a Dragon 64 and DragonDOS and I came across the same problem.

I found that the problem arose only after I had used the kill command in the *Teledisk* program, so the fault must be there and not with *Telewriter* and if the use of KILL within the program is avoided you can get more files on the disc. I usually average about 50 files alongside *Telewriter* and *Teledisk* on a single sided disc.

Another problem I find with the Dragon 64, but not a 32, is that if you wish to stop printing a file you have to turn off the printer as the print window on the 64 seems to be different to the 32. Other than these I have had no trouble with *Telewriter* and feel that it is an excellent word processor and hope the hints given here are of use to other users.

Robert Haigh
Gledholt Villas
63 Gledholt Bank
Gledholt
Huddersfield

Black and white case

THANK you for printing my screen dump in the November issue of *Dragon User*. I am afraid to say that I left out one important piece of information, ie that the pens in the printer have to be rearranged so that they are in the order green, black, blue and red. Without this users will get some very odd looking screen dumps.

Now for some information for fellow CoCo users. I don't know if you've noticed, but Tandy seem to have given up on the CoCo in favour of such trashy computers as the Spectrum (spit). As a matter of fact my two nearby Tandy shops have about two CoCo games, and about fifty Spectrum and Amstrad games. I have been unable to get any answer from Tandy about this even though I live near the UK headquarters. It seems the CoCo is going down the same road as the Dragon.

Finally, I have a complaint. Since going subscription, the quality of the printing in *Dragon User* has deteriorated, along with the spelling. The December column of *Adventure Trail* is full of spelling mistakes. I hope this is only a passing phase, as it detracts from the enjoyment of the magazine.

P.R. Marlow
50 Lime Avenue
Bentley
Walsall
W Midlands
WS2 0JP

THANKS, PJ. This is the jolliest letter I've received all week apart from my gas bill. Perhaps if we can get the black and white right we can graduate to getting our colour scrambled, what d'you say?

Those aren't spelling mistakes, they are typos. Adventure Trail must have slipped the net in December. Please accept our apologies. I must disagree about the print quality, apart from wealthy magazines with large circulations *Dragon User* has one of the best print standards about.

We are hoping to get more news on the progress of the CoCo, and in particular the CoCo Three in due course.

Every month we will be shelling out a game or two, courtesy of Microdeal, to the reader/s who send the most interesting or entertaining letters. So send us your hints and your opinions, send us your hi-scores and suggestions. Send us your best Dragon stories. What d'you think we are, mind readers?!

Dragon maths are the right answer

THANK you for the reply to my letter concerning the non-arrival of a recent *Dragon User*. I hadn't actually expected replies to the points raised, which were mainly inserted to fill up the page in the manner of casting bread upon the waters in the pious expectation that it would get all wet and sink!

Your point is taken about publishing lists of suppliers, and is one of the main reasons that I was so concerned about the missing DU. The point I was trying to make is that the shows are *not* likely to come to Ireland, and those of us in the wilderness rely on *Dragon User* as the only link to what is happening. To put it another way, you and the rest of the staff and contributors of DU are probably the only reason that the Dragon continues to exist as a viable computer. Since three years of consistently trying to have the school's Dragons replaced by a Nimbus network have consistently failed, the Dragons have to soldier on even though they are becoming more and more difficult to maintain, for example Peaksoft's joysticks are out of production, as I found when I tried to purchase them.

The kids are hard on joysticks, but when pupils in an 'inner-city' school badger their teacher to be allowed to come back after school to play computer games I must be doing something right. In case you don't find that surprising, I enclose screen dumps from two of the 'games' — they are standard 'joystick to right answer' type — in MATHS!

Keep up the good work.

Denis J. McCarthy
4 Summerville Terrace
Dalkey Co. Dublin, Ireland

MAGAZINES and special interest groups are rarely the only reason why something continues to exist, but they frequently provide a support system outside people's personal circle of friends and acquaintances and prejudices, even though the benefits may seem elusive at the time when it is not providing exactly what the consumer wants.

Any large group united by an interest is split by numerous factions with different angles. I recently saw a survey on what improvements Dragon User readers want from the magazine which was split very evenly between the three headings 'more OS-9/comms/Flex', 'more Basic' and 'less rubbish'. The even split is far more significant than the headings, because every month I get letters praising DU's articles on those and other subjects, and every month I get letters dismissing the identical subjects as 'rubbish'*

If a computer needs usefulness to justify its existence, then getting youngsters interested in maths should ensure the Dragon spiritual if not physical immortality.

**Dragon's Roar*, December 1987 Ed. Simon Jones

Disc bug trap

I am writing in response to the letter from Mike Hides. The problem described is not a fault in *Telewriter* but one of at least 15 bugs in SuperDOS E6. While I would like to say the only cure is to get a copy of DOSplus, the problem can be overcome by inserting the line

```
570 CLOSE
```

in the program TELEDISK BAS

Philip G. Scott
4 Badgerwood Drive
Frimley
Camberley
Surrey
GU16 5UF

Neophite whizzard

I SEE the cry is on again for more machine code. We are not all whizz kids, please keep DU as it is with something for everyone.

Only yesterday my son received a letter from one of his friends asking where he can get a Dragon, as he has had four, yes four Commodore 64s which have had to be returned as they would not work properly!

So there is always someone out there likely to start from scratch.

Could anyone in Essex who has a spare Dragon please contact my son's friend Emile Wilson, Upper Mead, Boyles Court, Dark Lane, Gt Warley, Brentwood, Essex CM14 5LL?

R J Rolph
254 Grasmere Way
Linslade, Leighton Buzzard
Beds LU7 7QB

A happy story, which we hope will have a happy ending and also a living, breathing example of points above. Why should the fact that some people want to do machine code mean that they are whizz-kids? You do not need genius to understand machine code, just a good guide and practice.

Emile may find he likes machine code, but if he wants to stick with shooting space invaders, I for one wish him all the best. Groups of people need a certain critical mass to provide resources which benefit them all, and if they retreat into a closed outlook that mass evaporates. Despite his anxious words, Mr Rolph seems like a good example of someone who doesn't mind sharing his Dragon world.

Projects please

I would like to add my name to the list of readers who are interested in hardware projects. I am sure that there is a lot that we can do to upgrade our Dragons. There must be people out there who produced lots of interesting add ons, unlike myself who given a soldering iron can fry chips in five seconds flat.

Perhaps somebody could astound Paul Harrison (December 1987 *64 Columns*) and work out how to connect up a 6845 VDU chip — it's only £9 from Maplin.

Most important of all, we must keep sharing our Dragon knowledge. See if we can put more fire into our Dragons in 88.

Alan Miller
6 Calder Gardens
Linslade
Leighton Buzzard
Beds LU7 7XE

If people ask for hardware long enough, we give them hardware. See page 14. This is not exactly an upgrade but it is interesting.

A is for Edit

I WOULD like to point out that if while using EDIT mode you accidentally delete the line then you can restart EDITing from scratch by pressing A (not while in insert mode).

James Bonfield
7 Water End
Wrestlingworth
Sandy
Beds SG19 2HA

Could be Norse

WE have just started a new Dragon magazine for all Scandinavian Dragon users. Write to use and we will send you the first issue free. The magazine is written in Norwegian.

Dragon Bladet
c/o Geir Hovland
Legdavegen
6943 Nausdal
Norway

Late one September

IF November comes can September be far behind?

Thank you for your letter and it does appear that the Royal Mail is at fault over the September issue. November is safely delivered but no September alas.

I hope you are successful in your recall of duplicated deliveries and if you are please remember me. Strangely enough I had no problems in July.

ACT Wilson
50 Goodacre
Orton Goldhay
Peterborough
Cambs
PE2 0LZ

Obscure dream

I note that as Pete Gerrard commented the month previously you have found my writing difficult to decipher last month as I note that my tip (*Dream saves faster* Letters December 1987) was mis spelt in places. To give you the correction I PRINT.

The start of symbol table address is held at locations &H5FBA and &H5FBB. You printed the & as an S! (*Easily done FR. You wrote &H5FBA twice in this letter instead of &H5FBB as the second location! Only a spot of razor-sharp detective work caught that in the nick of time. I hope Sherlock has it right this month — Ed*)

Also to exit from Dream one has to press BREAK then Q and *then* ENTER. You omitted the Q in the exit sequence (*It was a quiet day. No Qs at the exit. Groan. Sorry*) These two points may have left readers confused. Before reloading a Dream file which has been saved in this way the CLEAR address must be set to the value at which it was when the file was saved. This is so that the interpreter allocates this space for Dream so that loading the new file will not overwrite the stack. For those who don't know this is achieved by the command CLEAR N where N is the address of the highest memory you want BASIC to have access to; all higher RAM is reserved for Dream or whatever else (machine code, data etc).

Apologies for the lack of readability.

F R Ellehi
c/o 6 Clare Street
Halifax
West Yorkshire
HX1 2LF

THANK you for the prompt correction FR. One thing — what happened to the Word Processor? I hope your Dragon is not indisposed. My best wishes for its recovery if it is.

Words in reply

I WOULD like to reply to some of the points made in *Dragon User* November 1987 about my article *Three Little Words*.

To Mr Grayson, OS-9 allows suppression of LFs when a CR is sent. Stylo should not inhibit this. Some printers do not allow CR without LF, DIP switches or no.

To Mr Earnshaw, one of DragonDOS's few facilities is that it creates backup files — it is a shortcoming of the application that it does not support this.

I am unimpressed by references to some mainframe packages which I see as an attempt to mystify. I'm sure *Telewriter* is better than some out of date mainframe packages and how well do you think *Telewriter* compares to *Wordstar* or one of the more modern PC wordprocessors?

To Mr Rothery, OS-9's ROM seemed like a good idea to Tandy who use it with the CoCo3. You would lose nothing since additional commands would be on another input device. I can't imagine there would be no advantage in being able to power up in OS-9 rather than booting from disc. You yourself advocate the use of the RAM disc.

The point about the display is not what I prefer but whether colour set, foreground/background is user selectable.

I'm not impressed by an 80 column display that's in monochrome only or a RAM disc that drives a measly 64K and is not battery backed. But the price put me off from the start.

Telewriter right justify is undocumented — thank you for pointing it out.

The reference to MFREE? puzzles me in retrospect too. If any non OS-9 user is still unclear, Stylo reserves its own workspace automatically or as the user specifies.

64K of RAM is generally inadequate for smooth OS-9 operation in my view. The time you save with multitasking is lost in the large number of disc accesses that are required.

Roger Merrick
350 Gravelly Lane
Erdington
Birmingham
B23 5SH

Microdeal to leave the Dragon Market

MICRODEAL have now confirmed that they will be pulling out the the Dragon market as of 1st January 1988

They stress that the decision is financial and owing to diminishing returns and lack of suitable product to publish

Commenting on the lack of good new material John Symes said "This time last year we had fifty reasonable tapes to look at This year we only had ten There just isn't the material about any more.

The remaining stock of Dragon software has been taken aboard by Computape When asked if the list would be maintained a representative of Computape said that for the present they would concentrate on selling from stock but if the newer Microdeal games sold well enough they would consider maintaining them

Harry Massey of Computape recently told Dragon User that most of his business came from his company mailing list of about 5,000 Microdeal has confirmed that their Dragon mailing list is around 12,000 These figures are large enough to provide an active user base but it is possible that the bulk of prospective customers have already bought all that they require from the present list although Computape s low prices are bound to be attractive

On the subject of the Dragon John Symes said "The Dragon has been an exceptionally good machine for us and I would like to take this opportunity to thank all our supporters in the past and hope that in the future when they retire from Dragon computing that they move onto one of the machines we now publish for namely the Commodore Amiga and Atari ST

My thanks also go to all the staff at *Dragon User* who have kept this specialist journal and the Dragon market alive for much longer than expected

Dragon User thanks John Symes for his kind words But on behalf of many dedicated users may we add the hope that by the time the last of the Dragon users has retired Microdeal will be publishing games for machines as yet undreamed-of

Fourth Plus

PROGRAMMER P D Smith has completed a Forth compiler for the OS-9 operating system Based on the Fig-Forth standard the compiler includes extra commands from other systems to increase its power It compiles into pure machine code giving the fast running speed

As well as standard Forth commands it includes facilities for file handling allowing file handling without complicated block handling procedures There is a 30 page manual which has a section detailing the operation of the compiler for advanced programmers

The compiler is available from P D Smith at University Hall Birchwood Road Penyland Cardiff CF2 5YB price £14 95 The price includes software support and corrected updates if any bugs emerge

Power protection plug preferred

The SUPA 7 is a new power filter mains protection plug for computer equipment Rated at 7 amps the plug prevents voltage spikes from reaching equipment and filters out radio frequency interference Both these perils can be caused by electrical appliances switching as well as by more abstruse causes such as CB radio and atmospheric disturbances

The SUPA 7 can be used to protect word processors electronic tills telephone and telex systems videos and burglar alarms as well as computers The plug costs £14 95 plus VAT and 50p post and packing from Warwick Products 50 Marsh Wall West India Dock London E14 9XJ Tel 01 538 2535

OS-9 User group is here

An introductory sheet for the OS 9 User Group has recently fallen into the hands of *Dragon User* It has been warmly recommended by those who joined

The group was set up to help Dragon OS-9 users in 1985, and has since expanded into additional hardware it has a library with 46 volumes of public domain software much of which is in BasicO9 Pascal and C compatible at source level

The group s bulletin is published on disc called Newsdisk allowing them to send software as well as news and advertisements Members are supplied with a disc with which to correspond with the group

The group is run on a non-profit basis and membership is £10 per year For further details contact the OS 9 User Group (European) 4 Roseberry Court Llandudno Gwynned LL30 1TF

Disc drive deal

HARRIS Micro Software are proud to introduce a new range of fast modern high capacity dual and single switchable 40/80 track disc drives complete with SuperDOS cartridge for virtually the same price as their previous 40-track single sided drives

The drives manufactured by Viglen were originally designed for the BBC Micro and have the advantage of connecting directly to SuperDOS or CumanaDOS cartridges without modification

Prices for complete systems including cable manual cartridge and drive are £189 95 for the single 180 — 720K version and £289 95 for the 360 — 1440K version A deduction of £70 from the prices quoted covers the drive alone with cable.

Bob Harris can also supply a Viglen Dataduk signal splitter for joining two separate drives together if required The cable decides the priority of the drives avoiding the need for configuration at the computer and allows the drives to be switched at will The Dataduk costs £15

Snip database cuts down paperwork

Title *Diskbase*
Supplier Pulser Software 36 Foxhill High Crompton Shaw Oldham OL2 7NQ
Price £2 99 disc only

I CAN remember when my software collection comprised a handful of cassettes which were left lying around by my old '32 and all I had to do was quickly glance through them to find the program I wanted Now I have got three trays of cassettes and two boxes of discs and it used to take me hours to find a particular game or utility program

Having typed up my cassette collection into several databases only to find that they were full before I had entered everything and then they used to print out one under the other wasting three quarters of my paper I despaired of even starting to enter my entire disc collection

Pulser Software have come up with the answer in *Diskbase* written by Bill Tarrant On Sunday morning I loaded in the Help Screen and then followed with the Menu Starting with the Create screen I was prompted to input whether I was using one drive or two

As I have two drives I was able to leave my blank disc in Drive 2 while I inserted my program discs into Drive 1 After 20 minutes the information was on the working disc for all my 56 dull discs I then used the alphabetical sort which took another 45 minutes This may sound a long time but there were 647 different entries to be sorted not including Menus which are automatically left out by *Diskbase*

During this time you can go away and do other jobs while the computer carries on with the shell sort When sorted I printed out the full list which filled up three sheets of printer paper even though they were laid out over four columns This has to be better than twelve sheets with most databases Whenever I wish I can add to the database and there are numerous options open to me My printout is alphabetical with the name of the disc after it (you can use five characters including spaces to name your disc) but I would list out only Basic or only binary if I wished

A search can be made for any letters in it eg PAC would find SPACE WARS and PAC MAN or you can list what is on any particular disc. All listings can be made either to the screen or the printer It took me 1 hour 10 minutes to enter everything into the database and print out an alphabetical listing Other databases would need keying in and I do not know another database for the Dragon that could handle 216000 entries (600 entries for each letter and each number)

At £2 99 this has got to be a real snip and I would advise all disc owners to purchase it I would certainly not recommend piracy of it!!! Quite apart from the usual reason you are told on-screen to use the original only and there is a reason illegal copying will lead to the loss of your copy Do not worry no problem is incurred when you use the copy you have bought and anyway the original is duplicated on the other side of the Flippy on which it comes

This program has got to be worth five Dragons at this price

Mike Stott

A rolling stone gathers an Editor

Title *Boulder Crash 2*
Supplier Giant Soft 16 Moorcroft Road Sheffield 10 4GS
Price £3.50

QUITE a while ago I bought *Boulder Crash* from Blaby Computer Games and soon afterwards won a copy of **Stone Raider 2** in a competition These turned out to be virtually the same game although myself and my two sons preferred the Blaby version For those of you who do not know these games, the idea is to collect diamonds from around the screen of which you can only see a small part at any one time When you have collected a set number you can then go through the Exit to the next screen Boulders fall when you remove the earth under them and these can crush you if you are not careful On some screens you need to drop the boulders onto the Flappers who chase you to turn them into diamonds although this does not work with the Ghosties Beware also of the Slime which grows and tries to engulf you Power is also available at certain places and this can be fire — at pursuing creatures

Boulder Crash 2 is a screen-editor for the original so you will need both programs before you can start designing your own screens The instructions tell you to CLOADM the the kit program from Side B and that this will then copy and compile the original taking about 15 minutes but this is a once only task I cannot comment on the ease of the compiling as this review copy was already done for me When the revised program is loaded, there are two extra options from the main menu Pressing E takes you to the Screen Editor and X allows you to select infinite lives or any number from 1 to 100 Extra screens can also be loaded from Paul s tapes and I will comment on these later

When I first saw the Editing screen I nearly gave up as the graphics are abysmal but forget about that as once you go back to playing mode everything appears exactly the same as in *Boulder Crash* I have tried several of these edit your own screen ideas and I must admit that this is about the best I have seen although it is not really my cup of tea The instruction sheet that I received as very comprehensive and it is possible to design your own screen and play on it within a very few minutes but do not press reset if you are trapped somewhere as you will lose everything as I soon found out (I promise to read the instructions next time!)

Back to the extra screen provided with the program On the whole they are very good although several times my young seven-year old has trapped himself between two walls and two boulders and we have had to wait for the time to run out before we could start again due to not being able to reset Still I must admit that Paul Burgin s screens are certainly better than mine which are either too easy or just plain impossible Any screen you devise yourself can be saved out to tape so that you can play again whenever you wish and there is also a built in verify facility to ensure that they have been saved properly The keys used in the edit mode are very logical For instance D is for diamond W is for wall and SPACE is for a space

To sum up if you liked *Boulder Crash* and would like more screens or would like to design your own then I can recommend it If you have not seen the original then contact R & AJ Preston and you could very soon be buying this one as well If the price were under 2 50 I would give it four dragons but as *Boulder Crash* is not new I will give it three

Mike Stott

REBOOT

About the same time as we were publishing *Auto BOOT* last month another short listing emerged from the pile to shed some light on the non-operational DragonDOS BOOT Then a revision on that listing turned up Then a revision of the original turned up We aren t selfish so we ll share them

Mike Hides *presents his solution to the BOOT dilemma*

THE command BOOT allows one program on a disc to be loaded and auto-run without the need to enter its name Unfortunately no method is provided in the DOS to enable users to make use of this facility The following listing rectifies this and allows one machine code program on a disc to be BOOTed The listing works with Dragon DOS and SuperDOS but not with CumanaDOS

To use the program enter it using an assembler (written using Alldream) and omit the line numbers You need to enter the EXEC address of the program you intend to BOOT in line 1 and the file name in lines 33 and 35 Assemble the program and if free from errors place a blank formatted disc in drive 1 BREAK X (in Alldream) will execute the assembled program and write it to your disc Now copy the program you have decided to BOOT onto the same disc and all should now work

If you wish to BOOT a Basic program try the following modifications

a) delete lines 1 and 31
b) after line 30 insert the following three lines

```
JSR 33823
JSR 33773
JMP 33951
```

c) make sure you changed the BIN to BAS in lines 35 and 35

How it works

On entering the BOOT command the DOS loads sectors 3 to 18 of track 0 from the disc into memory starting at locations 9728. It then checks to see if the first two bytes contain the Ascii codes of OS and if so execution is automatically started at 9730

Lines 2 and 3 set the origin of the program to 9728 and put OS into the first two bytes Lines 4 to 8 do a CLS and put the title on the screen The next part is to lower the program in memory so that it does not clash with the program it will be BOOTing Lines 9 to 15 accomplish this movement The filename is then moved into location 166 and LOADED using the routine at $D4CE (lines 16 to 21) Finally lines 22 to 31 reset a Ram hook ensure that the warm restart vector is correct switch off the disc drive motors and jump to the execution address of your program

The final part of the program (lines 38 to 51) writes all the above onto track 0 of the disc starting at sector 3 The label @START tells the assembler where to begin the execution of this part of the program

Listing one

```
1   EXADR   EQU    *EXEC ADDRESS
                   *OF YOUR PROGRAM
2           ORG    9728
3   START   FCC    /OS/
4           JSR    $BA77
5           LDX    #$5D8
6           STX    $88
7           LDX    #TITLE 1
8           JSR    $90E5
9           LDX    #PROG
10          LDY    #$1DA
11  LOOP    LDA    ,X+
12          STA    ,Y+
13          CMPX   #PRGEND
14          BLS    LOOP
15          JMP    $1DA
16  PROG    LEAX   NEXT,PCR
17          STX    $168
18          LEAX   FNAME,PCR
19          LDA    ,X
20          STX    166
21          JMP    $D4CE
22  NEXT    LDX    #$D988
23          STX    $168
24          LEAS   2,S
25          LDA    #$55
26          STA    $71
27          LDX    #$C7D6
28          STX    $72
29          LSX    $8D8
30          STX    $FF48
31          JMP    EXADR
32  TITLE   FCC    /  BOOTING/
33          FCC    /FILENAME.BIN/,0
34  FNAME   FCC    34
35          FCC    /FILENAME.BIN/
36          FCC    34,0
37  PRGEND  FCC    0
38  @START  CLR    238
39          LSA    #3
40          STA    237
41          LDX    #9728
42  SAVE    PSHS   X
43          STX    238
44          JSR    49409
45          PULS   X
46          LEAX   256,X
47          INC    237
48          LDA    237
49          CMPA   #19
50          BLO    SAVE
51          JMP    33649
```

Dummy run

The previous program which allowed the use of the BOOT command to LOAD and RUN one file on a disc codes has a potential problem this is how to prevent the DOS writing over the code which resides on track 0 sector 3 of the disc. In the May 1987 issue Paul Dagleish gave details of how the directory track is organised and using this information the code can be protected

The method is to insert a dummy entry which occupies the appropriate part of the disc and cannot be deleted After putting the BOOT code onto an empty disc run the Basic program in **listing two**

Paul Dagleish described how the directory track was arranged into slots 25 bytes long starting at sector 3 on tracks 16 and 20 This program writes directly to these tracks and sets up a file called leave ALO NE! which appertains to occupy the appropriate position (track 0 sector 3)

Line 30 reads the disc directory track using the SREAD command and stores the current information in two string variables C$ and B$ Line 40 removes the information for the first directory slot from C$ Lines 50 to 110 read in the information for the dummy file from the DATA statements and a checksum is used to see if everything is OK It is now vitally important that only the disc with the BOOT information is in drive one otherwise when the rest of the program is run directory will be irretrievably altered Lines 120 to 150 give you the chance to abort Finally the SWRITE command is used in lines 1280 and 190 to write the information onto the disc Try DIR to see the result This dummy file cannot be deleted by the normal DOS commands and should protect your BOOT code

Listing two

```
10 CLEAN 400
20 A$ = "":B$ = "":C$="":CS = 0
30 SREAD1,20,3,C$,B$
40 C$ - NID$(C$,26)
50 FOR A = 1 TD 25
60 READ X$
70 A$ = A$ + CHR$(VAL("&H" + X$))
80 CS = CS + VAL("&N" + X$)
90 NEXT A
100 IF CS<>1203 THEN PRINT" DATA ERROR ":END
110 A$ = A$ + C$
120 CLS:PRINT" ARE YDU READY TO CONTINUE?"
130 PRINT:PRINT"ENTER Y or N"
140 Q$ = INKEY$:IF Q$ = "" TNEN 140
150 IF Q$<>"Y" TNEN END
160 PRINT:PRINT"PUT DISK WITH BOOT
    FILE IN","DRIVE 1 AND PRESS ANY
    KEY"
170 EXEC41194
180 SWRITE1,16,3,A$,B$
190 SWRITE1,20,3,A$,B$
200 DATA 02,6C,65,61,76,65,41,4C,4F
210 DATA 4E,45,21,00,02,01,00,00,00
220 DATA 00,00,00,00,00,00,FF
```

Julian Osborne has leapt, quite independently, to the same conclusion

SINCE the publication of my article *Auto BOOT* on the BOOT command in the October 1987 issue of *Dragon User* it has been noticed that under certain circumstances the BOOT routine causes problems where it occupies the first sectors of track 0 and can be over written by the DOS saving a program onto these sectors — not an ideal situation!

To get around this problem the following program de allocates the first 16 sectors of track 0 by altering the directory bit map to say that these sectors are occupied.

The procedure to set up a BOOT disc would then be

1) DSKINIT a fresh disc
2) Run the program given in **listing three** (DIR should then show 171008 free bytes)
3) Run the BOOT program from last month's article. This should then create the BOOT code on track 0 as normal but it will prevent DOS from using track 0 for any more programs. This routine works because SREAD and SWRITE do not refer to the directory contents when reading or writing to the disc.

The bit map for DragonDOS resides on track 20 sector 1 (for double sided 80-track drives it also takes up track 20 sector 2) and it represents the free space on a disc as follows.

Each of the bits in sector 1 of track 20 represents a disc sector: the first bit represents track 0 sector 0, the next bit represents track 0 sector 1 and so on.

If a bit is set to 1 then DragonDOS assumes the sector is free and if it is set to zero then it assumes that the sector is in use.

The RESERVE program reads the bit map into two strings (A$ and B$) and using the RIGHT$ function removes the first two bytes (set as 255 decimal which is 11111111 in binary). Then two null bytes (0 decimal or 00000000 binary) are added to the front of the string and the string is then written back to the Directory track 20 and also to track 16 (the reserve directory track). The two null bytes represent 16 bits which in turn represent the first 16 sectors of the disc (track 0 sectors 1 to 16) thus fooling DragonDOS into believing that they are occupied.

My apologies for not spotting this problem sooner but hopefully this program (and explanation) should solve it!

Anyway, as before I am willing to attempt to answer any queries about either the BOOT routine or the RESERVE program which may arise. My address is 8 Helston Road, Nailsea, Bristol BS19 2UA.

Listing three

```
10 REM RESERVE BOOT SECTORS ON DISK
20 REM RESERVES SECTORS 1 TO 16 ON TRACK 0
30 CLEAR 1000
40 SREAD 1,20,1,A$,B$ : REM READ BIT MAP SECTOR
50 C$=RIGHT$(A$,126) : REM CHOP OFF FIRST 2 BYTES
60 A$=CHR$(0)+CHR$(0)+C$ : REM REPLACE WITH 2 NULL BYTES
70 SWR1TE 1,20,1,A$,B$ : REM WRITE BACK TO BIT MAP ON DIRECTORY TRACK
80 SWR1TE 1,16,1,A$,B$ : REM WRITE BACK TO BACKUP DIRECTORY TRACK
90 CLOSE
100 DIR : REM SHOULD SHOW 171008 FREE BYTES REMAINING
```

BREAKing the '64

Martyn Armitage stops the 64's printer in its tracks

IN Pam D'Arcy's article *Epson Made Easy* in the February issue of *Dragon User* she mentions the fact that the Dragon 64 will not allow the PRINTER routine to be interrupted by pressing the BREAK key as on the 32. She also says that the bug hasn't been mentioned to date. Well, unfortunately the problem isn't a bug, but more of a case of poor forethought by the programmer. In the 32 ROM the BREAK key is checked for quite regularly and if pressed a return to basic is the result. When the ROM for the 64 was written an extra routine was thrown in. This extra routine simply looks at the state of the keyboard and if any key is pressed then the routine goes into a loop waiting until the keyboard is clear, ie no key pressed. Pressing any key including the BREAK key will cause the machine to wait for the key to be released. Once the key is released then a check is made to see if the BREAK key has been pressed and if it has then the usual return to Basic will ensue. Well then, why is it impossible to abort output to the printer? The answer is as said previoulsy, poor forethought. Between the wait while key pressed routine and the check for BREAK routine there are approximately 40 machine code instructions to be executed at an average of four microprocessor cycles per instruction. This gives a period of about 160 mpu cycles or approximately 180 microseconds (0.000180 seconds), not a lot of time to either release and re-press the BREAK key or to hit the BREAK key at the right time. Too early and the wait routine is entered, too late and the BREAK check is missed.

The problem is very infuriating, as the only way out is to wait for the LLIST etc to end naturally or press the RESET key. My answer to the problem is to put the computer into map 1 (all RAM), copying the ROM across and then to patch out the wait while key press routine. The short machine code program that follows accomplishes this, and it also copies the CARTRIDGE memory (&HC000 — &HEFFF) so that should a disc drive or cartridge be present then it can still be used as normal.

The machine code generated by the assembler listing sits in the first page of graphics if Dragon Dos is present, or the second graphics page if the Dos isn't present. The code is position independent and so can be relocated anywhere in memory if you should need to do so, for example if you have something in the graphics memory that you wish to preserve.

If you don't own an assembler (shame on you) then the basic listing will install the machine code which will then require saving, the addresses being:

Start &HC00
End &HC34
Exec &HC00

Remember that when the Dragon 64 is in map type 1 (all RAM) pressing the reset button causes the machine to revert to map type 0 (RAM and ROM) and doing so will cause the patch to be inoperative. If you do happen to press the reset button while in map type 1 you can type POKE &HFFDF,1 (ENTER), which will put the Dragon into 64k RAM mode and the patch should be intact. If not then you will have to reload the program and execute it.

Listing one

```
                        ORG     $C00
                        PUT     $C00
0C00     3401           PSHS    CC       SAVE CC REG.
0C02     1A50           ORCC    #$50     DISABLE IRQ'S
0C04     B68000         LDA     $8000    FIRST ROM BYTE
0C07     3402           PSHS    A        SAVE 'A
0C09     7A8000         DEC     $8000    ALTER FIRST ROM BYTE
0C0C     B68000         LDA     $8000    GET FIRST ROM BYTE
0C0F     A1E0           CMPA    ,S+      COMPARE WITH OLD VAL
0C11     2612           BNE     RAM      NOT SAME=MAP1 (RAM)
0C13     8E8000         LDX     #$8000   START OF ROM
0C16     B7FFDE   ROM   STA     $FFDE    SET MAP 0 (ROM)
0C19     A684           LDA     ,X       GET ROM BYTE
0C1B     B7FFDF         STA     $FFDF    SET MAP 1 (RAM)
0C1E     A780           STA     ,X+      STORE IN RAM, INC X
0C20     8CFF00         CMPX    #$FF00   LAST ROM BYTE?
0C23     26F1           BNE     ROM      NO SO GET NEXT BYTE
0C25     867E     RAM   LDA     #$7E     OPCODE FOR JMP
0C27     B78000         STA     $8000    RESTORE RAM BYTE
```

```
0C2A    8EBCFD    LDX     #$BCFD    SEND TO PRINTER ADDR
0C2D    B7BECD    STA     $BECD     OPCODE FOR JMP
0C30    BFBECE    STX     $BECE     PATCH OUT WAIT CODE
0C33    3581      PULS    CC,PC     RESTORE IRQ'S RETURN
```

Listing one cont

Listing two

```
10 CLS
20 FOR I=&HC00 TO &HC34
30 READ A$
40 A=VAL("&H"+A$)
50 POKE I,A:CS=CS+A
60 NEXT
70 IFCS<>6883 THEN PRINT "ERROR IN DATA":END
80 PRINT"CODE INSTALLED...":PRINT"REMEMBER TO SAVE IT"
90 DATA 34,1,1A,50,B6,80,0,34,2,7A,80,0
100 DATA B6,80,0,A1,E0,26,12,8E,80,0
110 DATA B7,FF,DE,A6,84,B7,FF,DF,A7,80
120 DATA 8C,FF,0,26,F1,86,7E,B7,80,0
130 DATA 8E,BC,FD,B7,BE,CD,BF,BE,CE
140 DATA 35,81
```

Winners and Losers

Every month Gordon Lee will look at some prize programming

THE competition was not in itself difficult although many readers were tripped up by minor problems. To quote Phil Sapiro: A very short program is all that is necessary ... and to prove the point listing one was what he attached. Paul Weedon worked along similar lines (listing two). By taking the digits in groups of five (still within the computer's mathematical capability) the running time was effectively reduced five-fold.

Almost without exception all of the winners adopted a version of these programs. However this month the non winners also came up with a number of surpnses. An identical 5231772315 from Randy Longshore and E A Newman. The explanation was simple! The programming and mathematics were perfectly correct, but they had both counted the number of digits from the decimal point end and not from the left. A technical knockout!

Surpnses also from Dave Lardner and Denis O'Mulloy who submitted 4787997625 and 5267797874 respectively. One is almost the reverse of the other. Closer examination reveals that the first five digits of Mr Mulloy's number reversed is the same as the first five digits of the answer while the last five digits of Mr Lardner's number is almost the same as the start of the answer. I suspect that the different digit is just a copying error and that both are five digits too soon in the series. Remember that though there are 34542 ones in the division there are only 34537 digits in the result. This difference of five is due to *six* ones needing to be called before the first digit of the answer appears – that is the division will begin 111111/34543. The number of digits in the answer is always five short of the number of ones.

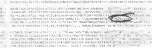
A good safeguard against this sort of error is to print out selected portions of the result. I have compiled the result into lines of 70 characters. The digits shown are from the beginning, middle and end of the full answer. To determine where to find a specific digit divide the number of the digit by 70. The 20001st digit is located by

20001/70 = 285 remainder 51

The required digit (which begins the sequence of our answer) is therefore found on line 285 at the 51st position.

Listing one

```
10 CLS:D=34543:Q=111111
20 FOR N=1 TO 20000:I=INT(Q/D):R=Q-D*I:Q=10*R+1:NEXT
30 FOR N=1 TO 10:I=INT(Q/D):R=Q-D*I:Q=10*R+1:PRINTI;:NEXT
```



[illegible]

Listing two

```
10 CLS
20 X=34543:Y=1111111111
30 FOR I=1 TO 20010/5
40 Z=Y/X
50 Y=VAL(STR$(Y-(INT(Z)*X))+"11111")
60 IF I>20001/5 THEN PRINT INT(Z);
70 NEXT I
```

Pamcodes

Part three of ***Pam D'Arcy's*** *introduction to machine codes*

AS listings get longer the chances of my incorrectly copytyping values from assembler listings increases the risk of errors so although I was trying to keep listings non-specific I am now supplying assembler listings with articles Lines commencing with an asterisk and the Comments following the semi colons to the right of source code lines are my assembler s equivalent of Basic s REM statements — they are for the programmer s benefit only and are ignored when the machine code is generated by the assemble process

Listing one

```
5001                  * LISTING 1
5001
5001
5001                  * JMPNOLFT (FILENAME)
5001                  *
5001                  * IGNORE LEFT ARROW KEYPRESS
5001                  *
5001                  * USING DREAM ASSEMBLER
5001                  * AFTER CLEAR200,&H5000
5001
5001 BDBA77    GO     JSR   $BA77    ,CLS
5004 CC0500           LDD   #$500    ,PRINT@256
5007 DD88             STD   $88
5009
5009 BD8006    GETKEY JSR   $8006
500C 27FB             BEQ   GETKEY
500E 8108             CMPA  #$08     ,LEFTARROW
5010 27F7             BEQ   GETKEY   ,YES-IGNOR
5012 BD800C           JSR   $800C    ,NO DISPLY
5015 810D             CMPA  #$0D     ,<ENTER>
5017 26F0             BNE   GETKEY   ,NO   CONT
5019 39               RTS
501A
```

Listing one (JMPNOLFT) covers the first requirement of last month s workout and **listing two** (JMPCHK) the second requirement As you can see in both instances the only amendment to the original PRINT@256 routine is the insertion of a few lines of comparison code between lines 30 and 40.

If you are struggling with references to ascii values for keypresses there is a list of them in appendix A of the manual supplied with the Dragon Capital letters are the decimal values >64 With the Editor still telling me to cut down on length I shall not include further comment on the code but please please do write in if there is something that you need to have explained further

CoMPare does not alter the two items being compared The result of the comparison causes various bits or flags of the condition code register (CCR) to be set (made=1) or unset (made=0) enabling different program paths to be followed depending on the result by using the conditional branch instructions.

So far only BEQ and BNE have been used in examples depending on whether the value in the register being compared is EQual or Not Equal to a fixed value in the operand column of the source code A slightly different case is where we immediately follow

GETKEY JSR $8006

with a BEQ instruction The Jump to SubRoutine instruction itself does not affect the condition code register This is a case where the ROM routine sets up two items of information (or return parameters) that we can use

— the Ascii code of a keypress if any else null ($00) in register A

— the condition code register to reflect the contents of register A that is a BEQ will be obeyed if there was no keypress (Register A=zero=$00) a BNE will be obeyed if Register A does contain a keypress value

Listing two

```
5001                  * LISTING 2
5001                  *
5001                  * JMPCHK (FILENAME)
5001                  *
5001                  * ERASE USING LEFT ARROW BUT DO
5001                  * NOT ERASE PRE START OF INPUT
5001                  *
5001                  * USING DREAM ASSEMBLER
5001                  * AFTER CLEAR200,&H5000
5001
5001 BDBA77    GO     JSR   $BA77    ,CLS
5004 CC0500           LDD   #$500    ,PRINT@256
5007 DD88             STD   $88
5009
5009 BD8006    GETKEY JSR   $8006
500C 27FB             BEQ   GETKEY
500E 8108             CMPA  #$08     ,LEFTARROW
5010 2607             BNE   DISKEY   ;NO-DISPLY
5012 9E88             LDX   $88      ;CURR CURS
5014 8C0500           CMPX  #$0500   ,POS @256
5017 27F0             BEQ   GETKEY   ,YES IGNOR
5019
5019 BD800C    DISKEY JSR   $800C    ,DISPLAY
501C 810D             CMPA  #$0D     ,<ENTER>
501E 26E9             BNE   GETKEY
5020 39               RTS
5021
```

As with IF statement in Basic if the conditions are met the instruction (ie branch) is carried out otherwise the program path continues with the instruction following the unsatisfied conditional branch instruction

Plus or minus

Signed and unsigned conditional branch instructions were mentioned last month Having just written a piece on it it was made to look very complicated and I realise that I was not supporting it with an example anyway as I want to make a start on adapting position dependent code to run on your own systems I am therefore holding it in abeyence for an appropriate time

Adapting code

A quick glance through the four Dragon machine code (as opposed to general 6809) books that I have show that most use position dependent rather than relocatable (or position independent) code in examples The instant give away is if one sees JSR and JMP rather than BSR and branch instructions excepting JSR/JMP to Basic ROM routines which are in our programs a special and (unless you are interested in running in Dragon 64 mode) acceptable case

The item that particularly caught my eye for our first example is *The Yellow Blob* from Jones and Cowsill s *Dragon Machine Code* (Shiva Publishing Ltd) but first consider the very simple examples of **listings three to five**

In **listing three** (JSRFRED) JSR (Jump to SubRoutine) generates the extended mode of instruction = an actual fixed address at addresses $5000 and $5009 respectively for FRED and HARRY Thus barring strokes of pure luck where locations $5008 and $5009 just happened to contain $39 at the time the program will only execute correctly if always loaded at its assemble address of $5001 If for example it was a machine code subroutine that we wished to call from a large Basic program the memory $5001+ would probably be occupied by Basic program and the machine code subroutine would need to be loaded and executed beyond the end of the Basic and its variables space probably around the $7000 area If the program was loaded at say $7001 using a load offset of $2000 (CLOADM"JSR FRED &H2000) the program still jump to addresses $5006 and $5009 to execute subroutines FRED and HARRY rather then a few bytes beyond the start of the program

There are two ways of tackling such machine code Even though one cannot sometimes assemble code directly in the area that one wants the machine code to operate from (for instance depending on which version is being used the Dream

Listing three

```
5001                * LISTING 3
5001                *
5001                * JSRFRED (FILENAME)
5001                *
5001                * EXAMPLE OF POSITION DEPENDENT
5001                * MACHINE CODE
5001                *
5001                * USING DREAM ASSEMBLER
5001                * AFTER CLEAR200 &H5000
5001
5001
5001 BD5008         GO      JSR     FRED
5004 BD5009                 JSR     HARRY
5007 39                     RTS
5008
5008 39             FRED    RTS
5009
5009 39             HARRY   RTS
500A
```

Listing four

```
5001                * LISTING 4
5001                *
5001                * JSRFRED2 (FILENAME)
5001                *
5001                * EXAMPLE OF POSITION DEPENDENT
5001                * MACHINE CODE
5001                *
5001                * USING DREAM ASSEMBLER
5001                * AFTER CLEAR200,&H5000
5001
5001
7000  7000                  ORG     $7000
7000                        PUT     $5001
7000
7000 BD7007         GO      JSR     FRED
7003 BD7008                 JSR     HARRY
7006 39                     RTS
7007
7007 39             FRED    RTS
7008
7008 39             HARRY   RTS
7009
```

assembler itself occupies $5E00+ in memory) the assembler may contain the option to generate code as if it were in the required area. Using Dream this can be achieved using the ORG (=ORiGin) and PUT directives. **Listing four** says generate code as if it is positioned at $7000 (ORG) but for the assembler place it in its normal workspace ($5001). If the PUT had been omitted the generated code would be placed directly into address $7000+ corrupting the Dream program with indeterminate but no doubt catastrophic results. ORG and PUT are what are known as assembler DIRECTIVES. They are not machine code instructions but the assemblers own means of allowing programmers to give it command and this is one of the areas of greatest difference in assemblers.

Having assembled the code in listing 4 that code is now fixed to execute successfully only from address $7000 in the machine. Just before I leave that, as this may well be an area that newer machine coders may have difficulty understanding with their assemblers, a word on saving machine code assembled in a different place from where it will run (at least for Dream assemblers, but perhaps it is also appropriate to other assemblers).

After assembly the generated code is actually sitting at address $5001+ in the machine. Its length is $7000-$7008 inclusive (first and last byte generated in the listing) = 9 bytes. It then needs to be saved with the likes of

CSAVEM "JSRFRED2" &H5001 &H5009 &H5001

but will only execute successfully if reloaded using an offset to take its start address to $7000 ($7000-$5001= $1FFF). Once reloaded at address $7000 (CLOADM JSRFRED2 &H1FFF) it can be further saved from that position in memory to avoid having to use offset for future loads, eg

CSAVEM FRED7000 &H7000 &H7008 &H7000

If your assembler contains facilities to ORG/PUT generated code this technique can be applied to any source code coded to occupy an area not able to be assembled directly into because of your assembler occupying that area.

This however is not necessarily the best solution to such position dependent code as there may be very good reasons for wishing to change its load position another time. One can amend the source and reassemble for a new load position but a much better solution would be to get rid of its position dependency altogether. The above is obviously a very simple example but by replacing extended mode JSR and JuMP instructions with relative branch instructions such instructions instantly become position independent or relocatable. **Listing five** (BSRFRED) replaces JSR instructions with BSR (Branch to SubRoutine) and the code generated is such that wherever the routine is loaded (barring corrupting Basic or DOS workspace!) it will function correctly.

Listing five

```
5001                * LISTING 5
5001                *
5001                * BSRFRED (FILENAME)
5001                *
5001                * EXAMPLE OF POSITION DEPENDENT
5001                * MACHINE CODE
5001                *
5001                * USING DREAM ASSEMBLER
5001                * AFTER CLEAR200,&H5000
5001
5001
5001 8D03           GO      BSR     FRED
5003 8D02                   BSR     HARRY
5005 39                     RTS
5006
5006 39             FRED    RTS
5007
5007 39             HARRY   RTS
5008
```

The yellow blob

Jones and Cowsill's *Dragon Machine Code* looked interesting enough from a quick perusal in a bookshop for me to add it to my collection. However the more I look at the example that I have selected the more I wonder how beginners to machine code may have coped with the sections before the book moves on to providing actual assembler listings. If you have struggled with trying to use the early listings as actual assembler source, the following should help throw light onto ways of amending it to make it produce the required object code and hence more understanding of the text. For those who have not got the book, working through the example may help you understand some of the things to look for in listings that you are experiencing similar difficulties with.

The program as it appears in the book (**listing six** — YELLBLOB) first caught my eye because of the use of JSR rather than BSR statements. However further examination yields even more valuable fodder for us to consider in trying to get other people's code to run on our systems.

The intention of the program is to be able to move a yellow blob around the text screen using the arrow keys. Pressing the break key terminates the program. I particularly like the way that this initial listing doesn't actualy include the code for the movement (subroutines UP, MDOWN, MLFT and MRGHT consist of an RTS). I often construct programs on this basis, even in Basic, to test that the skeleton works before putting on the flesh.

Monthly workout

Having said that the despite my intentions of not leaving you dangling mid-sentence as it were from month to month, I fear that my time is up. I will leave you then with the listing to ponder and perhaps see if YOU can get it going on your system. The program starts off by clearing the text screen (poking code #$60 to the text screen area ($400-$5FF) is the equivalent of writing ascii space characters to it using JSR $800C). The #$9F value is the text screen's yellow blob — see graphics

Listing six

```
*
* YELLBLOB  FILENAME)
*
* THE YELLOW BLOB - PAGE 56
* FROM 'DRAGON MACHINE CODE
* BY JONES&COWSILL (SHIVA)
*
* TYPED IN AS PER THEIR LISTING

6404    LDX    #0400    ;BE 04 00
        STX    6400     ;BF 64 00
        LDY    #01FF    ;10 BE 01 FF
        LDA    #60      ;B6 60
        CLR    6402     ,7F 64 02
        CLR    6403     ;7F 64 03
CLEAR   STA    ,X+      ;A7 B0
        LEAY   -1,Y     ;31 3F
        BNE    CLEAR    ;26 FA
        LDA    #9F      ;B6 9F
        LDX    6400     ;BE 64 00
        STA    ,X       ;A7 B4
KBD     JSR    8006     ;BD B0 06
        BEQ    KBD      ;27 EB
        CMPA   #5E      ;B1 5E
        BNE    DOWN     ,26 05
        JSR    UP       ;BD 64 53
        BRA    KBD      ,20 F2
DOWN    CMPA   #0A      ,B1 0A
        BNE    LEFT     ,26 05
        JSR    MDOWN    ;BD 64 54
        BRA    KBD      ,20 E9
LEFT    CMPA   #08      ;B1 08
        BNE    RIGHT    ,26 05
        JSR    MLFT     ;BD 64 55
        BRA    KBD      ;20 E0
RIGHT   CMPA   #09      ;B1 09
        BNE    BREAK    ;26 05
        JSR    MRGHT    ;BD 64 56
        BRA    KBD      ;20 D7
BREAK   CMPA   #03      ;B1 03
        BEQ    END      ;27 02
        BRA    KBD      ;20 D1
END     RTS             ,39
UP      RTS             ;39
MDOWN   RTS             ;39
MLFT    RTS             ;39
MRGHT   RTS             ;39
```

characters page of the Dragon manual Appendix A — and we should be familiar with the get keypress ROM call (JSR $8006) by now

The working visual result is not very exciting — green screen with yellow blob top left and the program simply sitting there until break is pressed — but the prospect of what you have achieved if you have used this listing (and not your own routine) particularly if your assembler is not geared to letting you generate code at $6400, should be very exciting indeed to you (in which case, you are probably beyond needing this series at all!) The only other thing that I will say is that if your assembler is like mine you will need to add in an awful lot of dollar signs to get it to generate code as this listing suggests (shown as comments in the source lines) Good luck!

Crossword

The third month of the Dragon Crossword We have the results from the first crossword now and the lucky winners whose correct entries were plucked from the editor s hat were Paul Simpson who wanted an up to date *Total Eclipse* which is now available from John Penn Software and Mrs H Clarke who gave us a list of choices most of em deletions! We will see what we can do — no promises

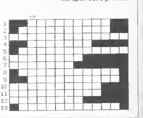
There will be a couple of free tapes from the Editor s Magic Bottomless Box for the first correct entries to reach us each month You can even try telling us which tapes you d like in an ideal world It all depends on what we can find

And you don t have to cut up your *Dragon User* either — heaven forbid! Entries can be written out on a photostat or a plain piece of paper as long as we can read em The ingenuity practiced by DU readers to avoid mutilating their precious paper work never ceases to astound

1 Plant this to raise the devil! (5 4)
2 Grinder run around the lines (4 6)
3 American President s burial place? (9 4)
4 He should not snatch! (7)
5 Tax for covert whirlpool (6 6)
6 I mounted trouble when the pit was closed (5 3)
7 Organise some resistance — don't sit on it (7)
8 Britain s black gold? (5,3,3)
9 Nasty Russians! (3 7)
10 If he did this slowly he could not be one! (5,5)
11 He looks for holes in a chart venue (4,6)
12 ET met a broker on South African journey (8)
13 The red beast visits a castle with a bishop (6,5)

by Terry and Derek Probyn

All this month's answers are names of Dragon software. When the crossword is complete, the column marked with an arrow will spell out a phrase.



Expert's Arcade Arena

HELLO and welcome to this month's straight laced column. Thanks for your letters but I want at least twice as many entries to the software survey as I've received — it's no contest with just my entry! (Editorial note: the Expert mocks you. Owing to the double time scheme operating here, you won't even have seen January's column at the time of writing.)

With the new year comes a new format for hints and cheats in the Arena. From now on they will have boxes round them so that they are easier to spot when you search through your *Dragon Users* looking for a particular tip. The first words in each box will tell you to which game it refers and whether it's a hint or cheat.

Got that? Good. Let's go!

Airball cheat

Load the game using Paul Burgin's program C (Arcade Arena September 1986) and co-ordinated with the line

20 POKE 32473, LIVES

where LIVES is an interger from 1 to 255

First off the top of the pile this month is this handy POKE for *Airball* which has been passed up and down the couytry and eventually reached me. The earliest origin of the POKE I know is Mr T Wilkinson. Thanks, Tom. Thanks also to James Bon[illegible] the first. Bad luck everyone who voted for *Airball* to be hacked in the survey but don't worry, your votes still count. The POKE's quite useful but how about a POKE to stop the ball deflating?

Sticking to *Airball* for a minute, I've a few pleas from people stuck with two screens. For one I provide a solution but the other I leave to you. The screen in question is described as SE SE SW SW SW NW SW which are ocmpass directions from the starting screen. It's a screen with a high wall and some vicious spikes, and is stopping people's progress with the game. Please send your solutions to the usual address.

Airball hint

To get past the scrteen which is SE SE, SW SW SE SE from the start screen, you must select eight directional movements. This enables you to jump into the room diagonally from the room before, and land on the two pedestals, not the deadly floor tiles. You must press two keys together (as well as bounce) to obtain this diagonal movement, and jump straight towards the bottom of the screen.

Everyone out there who's having trouble getting from one screen to another in time, don't forget the keys S and F will alter the spend of the ball.

James B also provides a few more POKEs, some of which have already been printer [illegible] me in the future. To show some appreciation of his work, perhaps someone can help him with a query. He would like to know what The Oracle is. No, not ITV's Teletext service, but a utility which seems to leave messages such as "The oracle's in RAM", "Enterbyte" and "Program loads too low in RAM" hidden within the coding of a large number of Microdeal games. Any ideas? You know the address.

Before I go, I've got here a few advance details for a couple of new games for 1988. Firstly *Crazy Foota 2*, which isn't available as I write, but will be by the time you read this. I haven't got copies of the new or old versions, but the letter says that it's now greatly improved, including different colours for each team, and options to allow a variety of games by changing the duration and speed of each game. The other new game I've been told about is the latest from Paul Burgin. This time he's stuck his revising claws into *Space Monopoly*. *Space Monopoly +* apparently includes a whole list of new features including holdings advice, controlling interests lists, joystick options, options to influence the creation of the universe and many others. The most interesting is an option to play against the computer from 0 to 4 players. I gather that Paul is now trying to work out how he can sell the game without upsetting Microdeal. (*Paul should be looking at the news page and having a quiet word with Harry Massey*)

Once again, the end is nigh. Apologies for a short column this month, but we all went a couple of days off at Christmas. I'll say Goodbye now, see you next month.

Adventure Contact

Adventure *Starship Destiny, Dungeon Destiny, Wild West Destiny*
Problem Hints wanted
Name Paul Iasikiewicz
Address 40 Sidlaw Avenue, Parr, St Helens, Merseyside WA9 2BQ

Adventure *Sea Quest/ Mystery of the Jaro Star*
Problem Where's the mirror? Where is the Ruby hidden on the ship?
Name Craig Dillon
Address 24 Glen Kinglass Road, Overton, Breenock PA16 9NW

Adventure *Galagon*
Problem How to get extra lives
Name Gary Hunt
Address 74 Ferndale Road, Leytonstone, London E11 3DN

Problem Pools predictor program needed, must be able to alter teams in divisions and send division results to printer
Name B Tozer
Address 10 Demond Ave, Beverley High Road, Hull. Tel 0482 443488.

Communication

Communication

Write down your problem on the coupon below (make it as brief and legible as possible) together with your name and address and send it to Communication, 12/13 Little Newport Street, London WC2H 7PP

Problem ....

....

....

....

Name ....

Address ....

....

....

Dragon drives direct

C.J. Walton describes an experimental interface project for DC motors

IN order to link the Dragon to motors to drive them it is necessary to use some form of interface There have been a number of articles in *Dragon User* detailing experiments in various aspects of interfacing and also on the production of interface units Such units generally have mechanical relays built into them or may be connected to suitable relays so that external device such as lights heaters motors etc may be switched on and off

It is usual practice in the area of robotics to use stepper motors to obtain accurate positioning of arms and grippers Such motors rotate in a series of very short but very precise angular steps Thus a typical motor may have 48 steps per revolution (7 5 degree step angle) and a more precise but more expensive type may have 200 steps (1 8 degrees step angle) Stepper motors are considerably more expensive than common DC motors, and require quite complex circuitry to control them The motors found in model railways model racing car systems in Fischer Technic kits and in Lego kits are all DC motors

This article presents experiments done to enable a measure of control to be obtained over the simple DC motors that are generally to hand Such control cannot be as precise as with stepper motors but being able to direct a buggy for example to behave rather like the classic Logo turtle is possible with the system

To simplify construction a ready built interface unit was purchased Electronic circuits were then built to enable the interface to run motors both forwards and in reverse Two motors each capable of independent control would then enable the buggy to be operated The interface unit obtained was that produced by NCJ Electronics of Hull whose address is included at the foot of this article It is the type 0101 and at the time of purchase it cost £14 95 inclusive It would be as well to enquire about cost before placing an order The interface was reviewed in the December 1984 edition of *Dragon User* It connects to the printer socket of the Dragon and has eight lines available to connect to external circuits The interface may be programmed in Basic using the PRINT statement or via machine code using the print subroutine B54A All the work done in this project involved Basic The interface uses a 74LS273 8-bit latch with a transistor on each output line to enable low current devices to be drive directly This interface does not contain relays and it is quite compact (about 60mm x 50mm)

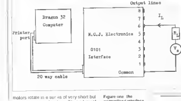


Figure one the generalised interface connections.

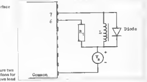


Figure two connections for an inductive load

Using the interface

Interface 0101 is built on a small printed circuit board and connected to the computer by a 20-way ribbon cable and plug To enable connections to be made easily to its outputs the board was mounted in a small circuit box with a hole made at the side for the ribbon cable The box was obtained from the local Tandy store (type 270-282 metal mini slope enclosure price £2 50 in a sale) and the eight line outputs and the common connection were taken to 4mm sockets which were mounted on the top of the case Leads with 4mm plugs were then used to connect the interface to the motor circuits which were also mounted in boxes with 4mm sockets

The transistor outputs are open-collector and work with inverted logic This can cause slight problems when driving motor circuits as will be detailed later The general connection and use of the interface is as shown in **figure one**

Note that the use of the Dragon s power supplies for external equipment is not advised Batteries or a separate (preferably stabilised) power supply should be used Under no circumstances should mains voltage be connected to any part of the interface

Initial tests with a number of DC motors without the interface indicated that currents greater than 200mA could occur Hence although the interface might be used to drive a small motor directly it was decided not to risk damaging it and to construct suitable motor drive circuits **Figure one** shows an external circuit connected between line 6 and the common connection Any or all of the lines could be so connected with the external power supply

Table one variable values to set outputs ON

Output line to be closed	1	2	3	4	5	6	7	8
value to be added to V	1	2	4	8	16	32	64	128

negative terminals all being returned to the common connection. Generally a single power supply would be used with the load for each line being taken to the positive side of the supply.

If the load is inductive, for instance a relay, then it is normal to include a diode across the load as in **figure two**. When the current in an inductor is switched off the back emf pulse can destroy the transistor driving current through the inductor. To protect the transistor the diode is included and carries the unwanted current pulse away from the transistor.

Figure two shows a non-inductive load R1 connected to output 6 and an inductive load such as a relay connected to output 7. A diode is connected across L and both loads are supplied from power source Vs.

Interface programming

To program the interface in Basic the format PRINT #2 CHR$ (V) is used. The semi-colon is necessary to prevent a carriage return code being sent after the desired command. The variable V is the value needed to set a particular series of outputs to on (or to act as a closed switch). **Table one** indicates the required values. For example to switch on lines 1, 3, 5 and 8, the value of V would be given by

V = 1+4+16+128=139

The program line would then be

PRINT # 2 CHR$(139)

The other lines would be left open. When a line is closed there is effectively a complete circuit from the external power supply through the load, through the line output and common connection back to the power supply. The interface acts like eight separate single pole single throw (single pole on off) switches. Any combination of the switches may be operated by the PRINT command. As it stands, the interface may be used to control a (small) motor directly by connecting the motor as the load L in **figure two**. Rather than risk damaging the interface by driving too much current through it, it is better simply to use the interface as a switching device and to operate the motor via an external transistor as in **figure three**.

An alternate circuit is shown in **figure four**. Here control of the motor is effected via the output line side of the interface. This becomes necessary if it is desired to operate a number of motors with the circuit relevant to a typical medium power NPN transistor. If the interface is switched off

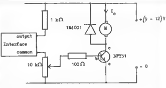


Figure three operating the motor through an external transistor

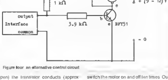


Figure four an alternative control circuit

(open) the transistor conducts (approximately Ic = 100mA) and the motor runs. When the interface is switched on (closed) the transistor base b is effectively shorted to the 0V line and the transistor is switched off (Ic = 0). The 1 kilohm resistor protects the interface from excess current when it is switched on (otherwise the power supply would be shorted across the interface).

Using a circuit similar to **figure four** a number of motors may be driven via the interface, each from one of the line outputs and with the transistor emitters all taken to the interfaced common line.

To operate the circuit of **figure three** we need to close the interface as a switch and thus apply bias to the transistor base to cause current to flow in the collector. If we open the interface, then there is no bias to the transistor and hence the BFY51 (a typical medium power transistor) will be off. **Listing one** gives the commands to switch the motor on and off ten times. Output line 1 is used. Thus the command

PRINT # 2 CHR$(1)

closes line 1 and opens all the other lines

PRINT # 2 CHR$(0),

or any other value for V will open all lines, including line 1.

The circuit of **figure four** has to operate in an opposite way to that of **figure three**. Here we need to open the interface switch in order to apply bias to the transistor. **Listing two** gives an example to achieve this. Again we use line output 1. By setting the value of V to 1 (line 10 of the listing) we close the interface switch, this switches the motor off initially. In line 50 V = 0. This opens the interface switch and causes the motor to run. Operation of the motor is

```
10 FOR I=1 TO 10
20 PRINT#-2,CHR$(1);: REM MOTOR ON
30 FOR T=1 TO 2000: NEXT T
40 PRINT#-2,CHR$(0);: REM MOTOR OFF
50 FOR T=1 TO 3000: NEXT T
60 NEXT I
```

Listing one command for ten switchings.

```
10 PRINT#-2,CHR$(1);: REM MOTOR OFF
20 K$=INKEY$
30 IF K$="R" THEN 50 ELSE IF K$="O"
 THEN 60 ELSE 20: REM <R> TO RUN,<O>OFF
50 PRINT#-2,CHR$(0);: GOTO 20
60 PRINT#-2,CHR$(1);: GOTO 20
```

Listing two command to open the interface switch

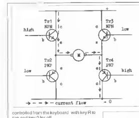


Figure live a circuit to operate the mater in either direction

controlled from the keyboard with key R to run end key 0 lor off

Since the remainder ol lhe circuits tested are of lhe type shown in **figure four** it will be necessary to open lhe interface swilch on a particular line if we wish lo lurn a lransistor on Sometimes we must have a number of lines open while olhers are closed Care must lhen be leken with the linal value of V used in lhe PRINT statemenl Both circuits **figure three** and **figure four** were assembled on TDec boards lor lest ng Resistor values may then be easily changed es may lhe transistors used Any kind of prototyping or breadboard will do the job

A major problem with circuits of lhe form shown in **figures lhree and four** is lhal both lypes can only switch a molor on or off When running it will do so in one direclion only The most verselile conlrol occurs when a motor can be swilched lo run in either direction This can be achieved by using two of the interface oulpuls lwo lransistors and lwo relays Suitable connection of lhe molor to the relays can lhen resull in motion in one direction conlrolled by one lransistor and motion in the olher direclion from lhe second transislor A more elegenl solution (and one which is cheaper and more reliable because mechanicel relays are avoided) involves using a lour lransistor bridge of H circuil

Figure five illuslrales lhe principle Two NPN and PNP lrensistors are used (ideelly lhey will be matched complimentary pairs) When lhe inputs lo Tr1 and Tr4 are sel high current llows throughl lhem and drives lhe motor M in one direction Tr2 and Tr3 are switched off by selling lheir inpuls low lhat is to lhe zero line

Il Tr1 and Tr4 are lhen sel low with Tr2 AND Tr3 set high current llows n Tr3 and Tr2 and lhe molor runs in the opposlive direction Setting all inputs low swilches all lhe transistors off and lhe motor slops Setting all lhe inpuls h gh would also switch lhe motor off but since some current may llow in the transislors il is preferable to swilch lhe motor off by setting all inpuls low

Figure six gives a lypical bridge circuit Component values are nol critical and il is worth experimenling on e Dec board before soldering up a final version

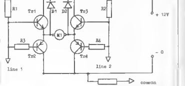


Figure six a basic bridge circuit

Resistors R1 to R4 should ell have the same value The 2N2905 is a PNP lransislor with similar cheracleristics lo the NPN BFY51 lype As before diodes (1N4001 lype) are used to protect lhe lrensislors when the molor is swilched off or reversed R5 is used lo protect lhe interface should lhere be a lault in lhe circuit allowing lhe full supply voltage across the interface line and common lerminals It will restrict lhe current in lhe interface to a safe level (lhat is more than 200mA)

Single motor drive unit

It was decided lo construct the circuil ol **figure six** on a small circuil board and enclosed il in a suitable box Tendy stores supply a variety ol plaslic and metal boxes a plaslic box 81mm x 51mm x 35mm was chosen This came complete wilh a board which had a regular erray of holes drilled in it and copper laminate connections lo lhe holes **Figure nine** gives lhe component layout on the board Small metal pins (obtainable from electronics suppliers) were inserted into the board and short lengths of stripped wire used lo link lhe pins to act as bus bars for lhe individual componenls Thus lhe componenl layout is direclly comparable wilh lhe circuil diagram of **figure eix** Lead connections lor the diodes and transislors are given in **figure eeven** and **figure eight**

Although diodes and transistors are fairly rugged mechanically and electrically lhey can be damaged by excess heat A useful hint when soldering semiconductors is to grip lhe lead between component body and soldering point wilh e pair ol long-nosed pliers To avoid lhe necessily for a lhird hand when soldering wrep a strong elastic band around the handles of lhe pliers so that they ere pulled shul When held open and placed onto the component lead lhey will grip it firmly withoul lhe need lo be held by hand The pliers also hold lhe componenl in place on the board if you carelully arrange lhings prior lo soldering Allow lhe heat to dissipale before removing lhe pl ers

4mm sockets were mounled on the box and w res connected lo lhese lrom lhe circuit board for lhe two lines common motor and power supply To assist n making connections lo lhe unit lhe following colour coding was used for lhe sockets — line inputs green common blue molor white power supply positive red power

Figure seven diode lead connections.

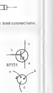


Figure eight transistor lead connections bottom view

```
5 REM SINGLE MOTOR CONTROL
10 PRINT#-2, CHR$(3J,' REM LINES I & 2 S
WITCHEO ON, MOTOR OFF
20 K$=INKEY$
30 IF K$='R' THEN 50 ELSE IF K$="L" THEN
 60 ELSE IF K$='O' THEN 10
50 PRINT#-2, CHR$(1];. GOTO 20: REM LINE
 I ON, LINE 2 OFF, MOTOR RUNS RIGHT
60 PRINT#-2, CHR$(2];. GOTD 20: REM LINE
 1 OFF, LINE 2 ON, MOTOR RUNS LEFT
70 REM LINE 1 HAS V=1,LINE 2 HAS V=2
```

Listing three a program to rnn the motor teft and right from the keyboard

supply negat ve black This single motor drive unit could then be connected to the interface on the one hend end to a motor on the other A Lego 12 volt DC motor was used to lesl the unit This look 0 26A when runn ng and driving some gears In another arrangement it took 90mA with less loading When the motor was off the circuil current dropped to 17mA With the motor running the voltage across one of the conducting transistors was about 1 5V so that the power dissipation in the transistor was some 350mW well below the transistor maximum (approx 7mW)

Listing three gives a short program to control a motor to rnn right and left from the keyboard To switch the motor off we need the V value to be such that both l nes are ON from the interface This sets the two bridge inputs low TO turn the motor on we want one input low and one high ie one line to be ON and the other OFF To have line 1 ON we set V=1 which automatically puts l ne 2 OFF TO have line 2 ON we set V=2 which puts line 1 OFF Any other value for V will put both lines OFF The program uses V=3 V values for other lines are given in **Table one**

If the motor runs left when the R key is pressed simply reverse the motor connections To control motor direction by the cursor keys replace the R and L of line 30 by CHR$(9) and CHR$(8) respectively

Double motor drive unit

In order to run two or more motors independently we must have a separate four transistor bridge circuit for each motor Each bridge circuit then requires two lines from the interface unit Two motors operating independently enables a var ety of two-motor devices to be controlled Examples could be two model trains (using separate power supplies to the two sets of tracks) and X-Y plotter or a buggy

The circuit for the double motor drive unit is simply the combination of two of the single motor units operating from one power supply **Figure ten** gives the circuit An extra component D5 is used to protect the circuit (with its eight transistors) from inadvertently connecting the power supply with the wrong polarity With no motor operating the quiescent current for a 12V supply was approximately 20mA Operating values of current would depend on the motors being driven (see the section on use with a Lego buggy)

Construction of the circuit was again on a pinboard this time sized about 140mm x 40mm (cut to slot into a circuit box of dimens ons 150mm x 80mm x 50mm obtained from Tandy) A diagram of the component layout is given in **figure eleven** As with the single motor unit holes were drilled into the circuit box to take 4mm sockets To avoid confusion during connect on I use the colour coding inputs green motor outputs white power supply positive red power supply negat ve black common connection to interface blue The transistors have maximum ratings as follows BFY51 Vce = 30V Ic = 1A Pdiss = 08W 2N2905 Vce = 40V Ic = 06A Pdiss = 06W When operating Vce is approximately half the supply voltage Continuous operation could cause overheating if a motor takes more than about 0 1A Heatsinks on the transistors may then be advisable The present unit has not been run for more than a few minutes continuously and the transistors barely appear warm to the touch with a 12V supply and 0 2A maximum current per motor

The double motor drive unit was used to control a model robot c arm built from a Fischer Technic kit (Model 30554 Computing) This uses two motors one to rotate the arm left and right (motor 1) and one to raise the arm up and down (motor 2) The first four lines of the interface were linked to the drive unit with lines 1 and 2 for motor 1 and lines 3 and 4 for motor 2 Motor 1 was controlled by keys R L and O (off) with motor 2 controlled by U D and S (stop)

To enable one motor to run whether or not the other was on at the time meant that some method of indicating the state of the motor was needed This was achieved by having indicators or flags in the program L1=0 means motor 1 is not running left L1=1 means that it is running left L2=0 means motor 2 is not running left L2=1 means that it is running left (or down in the case or motor 2) Thus every keypress sets an indicator

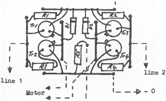


Figure nine a component layout for the circuit board

Tr1,3	BFY51	D1,2	1N4001
Tr2,4	2N2905	R1-4	1 kohm
		R5	100 ohm

Listing four gives lots of REMS to indicate what is happening in the program The V values are given in the PRINT statements and were selected so that whether motor 1 is running or not motor 2 can be switched on or off Likewise motor 1 can be opereted irrespective or the state of motor 2 The V values are listed in **table two** for this arrangement Since the interface has eight output lines a total of four separate motors each with its own power supply could be independently operated If each one s to be opereted independently of the others then suitable V values would need to be selected again with indicators to report the state of each motor Each motor would need one bridge circuit

One multiple motor application could involve a model railway Up to four trains could be programmed to run quite independently along separate tracks A suitable program could be devised which would switch particular trains on or off at certain times They could run forwards or backwards The Dragon could control a set

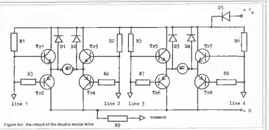


Figure ten the circuit of the double motor drive

timetable of train arrivals and departures from a number of stations for each train One could even simulate British Rail by having a random element built into the program so that each train suffered a number of late arrivals and early departures but exactly when would not be known!

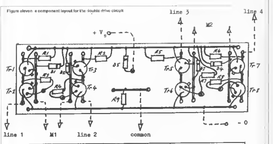
To add two more motors with an additional motor drive circuit for each one new V values would be have to be calculated **Table 3** shows the basic values in pairs for right and left rotation for each motor Combining these values as in **table two** would then be required A comparison between the tables should enable this to be done

The Lego buggy

The Lego buggy is assembled from a Lego kit (price about £23 ex carriage and VAT) containing two 4 5V DC motors Each motor may be used to drive one large wheel while a small rear wheel (undriven) provides a third contact to the ground Running both motors together in the same direction can produce forwards and reverse motion Keeping one motor stationary while running the other or running the two in opposite directions causes the Buggy to turn

It is emphasised by Lego that the motors should not be supplied with more than 4 5V otherwise they may burn out Initial tests indicated that the motors tend to take a fairly constant current Over the voltage range 2 to 4.5V a single motor took 0 2A while the two in parallel took 0.34A With a motor stalled (ie trying to drive against an obstacle) the current taken by a single motor rose to about 0 25A The stall current may rise up to 1A if the motor alone is without gears and wheels attached is prevented from running at maximum supply voltage With the Buggy motors connected to the motor drive unit a supply

Figure eleven a component layout for the double drive circuit



Tr1,3,5,7	BFY51	D1-5	1N4001	R9	100 ohm
Tr2,4,6,8	2N2905	R1-8	1 kohm	M1-2	d.c. motor

Listing four the control program with flags and REMs

```
5 PRINT#-2,CHR$(15); REM ALL 4 OUTPUTS
CLOSED: MOTORS OFF
10 L1=0: L2=0: R1=0: R2=0: REM MOTOR IND
ICATORS SET TO 0
15 CLS: PRINT:PRINT"     TO CONTROL MOTI
ON": PRINT"     USE THE KEYS"
16 PRINT:PRINT"     <R> MOTOR 1 RIGHT":
PRINT: PRINT"     <L> MOTOR 1 LEFT
17 PRINT:PRINT"     <O> MOTOR 1 OFF": PR
INT: PRINT"     <U> MOTOR 2 UP": PRINT:P
RINT"     <D> MOTOR 2 DOWN": PRINT: PRIN
T"     <S> MOTOR 2 STOP"
20 K$=INKEY$
30 IF K$="R" THEN 130
40 IF K$="L" THEN 140
50 IF K$="O" THEN 150
60 IF K$="U" THEN 160
70 IF K$="D" THEN 170
80 IF K$="S" THEN 180
90 IF K$="" THEN 20
130 R1=1: IF R2=1 THEN PRINT#-2,CHR$(5);
 ELSE IF L2=1 THEN PRINT#-2,CHR$(9); ELS
E PRINT#-2,CHR$(1);: REM SWITCHES MOTOR
1 ON RIGHT
131 GOTO 20
140 L1=1: IF R2=1 THEN PRINT#-2,CHR$(6);
 ELSE IF L2=1 THEN PRINT#-2,CHR$(10); EL
SE PRINT#-2,CHR$(2);: REM SWITCHES MOTOR
 1 ON LEFT
141 GOTO 20
150 R1=0: L1=0: IF R2=1 THEN PRINT#-2,CH
R$(7); ELSE IF L2=1 THEN PRINT#-2,CHR$(1
1); ELSE PRINT#-2,CHR$(15);: REM SWITCHE
S MOTOR 1 OFF
151 GOTO 20
160 R2=1: IF R1=1 THEN PRINT#-2,CHR$(5);
 ELSE IF L1=1 THEN PRINT#-2,CHR$(6); ELS
E PRINT#-2,CHR$(4);: REM SWITCHES MOTOR
2 ON RIGHT (UP)
161 GOTO 20
170 L2=1: IF R1=1 THEN PRINT#-2,CHR$(9);
 ELSE IF L1=1 THEN PRINT#-2,CHR$(10); EL
SE PRINT#-2,CHR$(8);: REM SWITCHES MOTOR
 2 ON LEFT (DOWN)
171 GOTO 20
180 R2=0: L2=0: IF R1=1 THEN PRINT#-2,CH
R$(13); ELSE IF L1=1 THEN PRINT#-2,CHR$(
14); ELSE PRINT#-2,CHR$(15);: REM SWITCH
ES MOTOR 2 OFF
181 GOTO 20
190 REM COMMANDS FOR ONE MOTOR ALLOW OTH
ER MOTOR TO OPERATE INDEPENDENTLY
```

Listing five the Buggy control program

```
5 REM BUGGY CONTROL
10 PRINT#-2,CHR$(15);: REM ALL 4 OUTPUTS
 CLOSED: MOTORS OFF
20 CLS: PRINT: PRINT"       TO CONTROL B
UGGY": PRINT"       USE THE KEYS"
25 PRINT: PRINT"       <F> FORWARD"
30 PRINT: PRINT"       <B> BACKWARD"
35 PRINT: PRINT"       <L> LEFT
40 PRINT: PRINT"       <R> RIGHT
45 PRINT: PRINT"       <S> STOP
50 K$=INKEY$
60 IF K$="F" THEN 160
70 IF K$="B" THEN 170
80 IF K$="L" THEN 180
90 IF K$="R" THEN 190
100 IF K$="S" THEN 200
110 IF K$="" THEN 50
160 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(5)
;: GOTO 50: REM M1 ON R, M2 ON R
170 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(10
);: GOTO 50: REM M1 ON L, M2 ON L
180 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(6)
;: GOTO 50: REM M1 ON L, M2 ON R
190 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(9)
;: GOTO 50: REM M1 ON R, M2 ON L
200 PRINT#-2,CHR$(15);: GOTO 50: REM BOT
H OFF
```

Listing six an alternative control program

```
5 REM BUGGY CONTROL2
10 PRINT#-2,CHR$(15);: REM ALL 4 OUTPUTS
 CLOSED: MOTORS OFF
20 CLS: PRINT:PRINT:PRINT"       TO CONT
ROL BUGGY"
25 PRINT: PRINT"       USE CURSOR KEYS"
30 PRINT: PRINT:PRINT:PRINT"       TO STO
P"
35 PRINT: PRINT"       USE SPACE BAR"
50 K$=INKEY$
60 IF K$=CHR$(94) THEN 160:REM F
70 IF K$=CHR$(10) THEN 170:REM B
80 IF K$=CHR$(8) THEN 180:REM L
90 IF K$=CHR$(9) THEN 190:REM R
100 IF K$=" " THEN 200
110 IF K$="" THEN 50
160 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(5)
;: GOTO 50: REM M1 ON R, M2 ON R
170 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(10
);: GOTO 50: REM M1 ON L, M2 ON L
180 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(6)
;: GOTO 50: REM M1 ON L, M2 ON R
190 PRINT#-2,CHR$(15);: PRINT#-2,CHR$(9)
;: GOTO 50: REM M1 ON R, M2 ON L
200 PRINT#-2,CHR$(15);: GOTO 50: REM BOT
H OFF
```

To run Motor 1	ON R	ON L	OFF
with Motor 2 OFF	1	2	15
ON R(U)	5	6	7
ON L(D)	9	10	11
To run Motor 2	ON R(U)	ON L(D)	OFF(STOP)
with Motor 1 OFF	4	8	15
ON R	5	9	13
ON L	6	10	14

Table two V values for Listing four

Motor 1		Motor 2		Motor 3		Motor 4	
R	L	R	L	R	L	R	L
1	2	4	8	16	32	64	128

Table three values for left and right rotation

	M1	M2	V
To move forwards F needs both motors ON R	R 1	R 4	5
To move backward B " " " ON L	L 2	L 8	10
To move left L needs M1 L and M2 R	L 2	R 4	6
To move right R needs M1 R and M2 L	R 1	L 8	9
To stop Buggy needs both motors OFF	1+2	4+8	15

Table four values to drive the Buggy

of 9V was needed to ensure that each motor received 4.5V or a little less when the Buggy was operating

The Basic program needed to control the Buggy s movements requires suitable values of variable V These will depend on how the motors are wired with regard to the polarity of the outputs from the motor drive unit It will be necessary to experiment a little here Viewing the Buggy from above and denoting the left hand motor M1 and the right hand motor M2 we may set up the table shown A motor running right R is deemed to be in the forward direction running left L is taken as backward/reverse

Note that in order to turn to L or R both motors are running Turns may be effected by stopping one motor and driving the other However if this is done the power supplied to the Buggy is halved and the single motor running has difficulty in driving the Buggy Hence the speed falls On the other hand by stopping one motor the Buggy will pivot about the point of contact of the stopped wheel This will give a more precise description of the path of the Buggy In this case the third and fourth lines of the previous table need modifying to

The Buggy control program given in **listing five** operates with both motors running for a turn Motion direction is selected by the F B, L and R keys with STOP by S When a new key is pressed the command to stop both motors is issued prior to starting the new motion In view of the speed of the Dragon s signals this command will have little effect However if a time delay is inserted after the command (in lines 160 to 190) the Buggy can be made to pause for as long as required This may be useful in some situations.

An alternate control program is given in **listing six** The Buggy will now respond to the cursor keys for direction control and is stopped by the space bar This may be found to be a more convenient method of control particularly if an extended series of direction changes is needed Again both motors operate on a turn

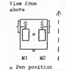
Having obtained control of the Buggy a felt tip pen could be attached to it (eg clamped by a clothes peg) and Turtle type graphics done on paper For true LOGO control one needs to adapt a suitable program (perhaps Mike Horden s *Mini LOGO* in *Dragon User* September 1986) or write one s own

To move left L needs M1 OFF and M2 R	OFF 1+2	R 4	7
To move right R needs M1 R and M2 OFF	R 1	OFF 4+8	13

Extra points

To stop the Buggy when it collides with an obstacle one technique uses microswitches at the front and rear of the Buggy These are in series with diodes and the motor as shown in **figure twelve** The switches should be of the normally closed type and when contact is made with the obstacle the switch opens and cuts off current to the motor via the relevant connector Two diodes and two switches are used because current has to flow in two directions depending on which way the motor is running

If more powerful motors are to be used the transistors used here will be overloaded In this case two transistors of the same type may be doubled up to form a Darlington pair as in **figure thirteen**



The Buggy viewed from above

Alternately it is now possible to buy devices containing one Darlington pair in a single package or multiple pairs in a single package Recently complementary transistor arrays have appeared One contains two NPN and two PNP transistors in matched pairs in a 14 pin DIL plastic package (500mW dissipation per transistor) This would be ideal for producing one motor drive unit circuit for say the Lego motor on a printed circuit board (but rather expensive as yet)

Finally please note that I have no business connection with NCJ Electronics the producer of the 0101 Interface Unit used for the experiments described in this article It was good luck in the first place that acquainted me with the interface and enabled me to devise the experiments for myself over a period of time

Component sources

The address of **NCJ Electronics** is 38 Murrayfield Road Chanterlands Ave Hull HU5 4DW

Two suppliers of electronic components to the amateur constructor are given below Others of whom the largest and best known is probably Maplin Electronics may be found in the pages of any electronics magazine such as Practical Electronics Electronics Today International Everyday Electronics Practical Wireless or Radio and Electronics World

Rapid Electronics Hill Farm Industrial Estate Boxted Colchester Essex CO4 5RD

Magenta Electronics 135 Hunter Street Burton-on-Trent Staffs DE14 2ST

The Lego Buggy can be obtained from **Economatics** Ltd Epic House Orgreave Road Handsworth Sheffield S13 9LQ **Unilab Ltd.**, Clarendon Road Blackburn Lancs BB1 9SS and **Magenta Electronics** It is also available in some large toy stores

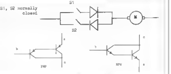


Figure twelve bi-directional switching to step the Buggy

Figure thirteen transistors doubled to form a Darlington pair

Parts lists

Single motor drive

2 x BFY51 NPN transistor
2 x 2N2905 PNP transistor
2 x 1N4001 1A 50V diode
4 x 1 kilohm 0 25W resistor
1 x 100 ohm 0 25W resistor

1 plastic box to suit
1 x pinboard to fit box
7 x 4mm sockets
Pins and connecting wire

Approx cost of unit £4 00

Double motor drive

4 x BRY51 NPN transistor
4 X 2N2905 PNP transistor
5 x 1N4001 1A 50V diode
8 x 1kilohm 0 25W resistor
(8 x TO5 package heatsink)

1 plastic box to suit
1 pinboard to fit box
11 x 4mm sockets
Connecting wire

Approx cost of unit £8.50

While Dragon User makes every reasonable effort to ensure that published projects are viable, it cannot be held responsible for any loss or damage arising from such projects These projects do not employ mains voltages directly and are not difficult to build but require care and attention to detail to achieve success If in doubt ask the advice of someone with greater experience of electronic construction Check that all the components you want to use are available from suppliers and the current prices before ordering

THE DRAGON AND TANDY SHOW

AT CARDIFF — WALES AIRPORT

on Saturday
27th April 1988
— 10.00–3.00

Adults £1.00 *Children 0 50p*

Don't miss the opportunity to meet the major retailers and to watch demonstrations by Dragon Users

Ample car parking Easy access
Regular bus service via M4
from Cardiff Centre Refreshments

If you would like to take part as a demonstrator, showing how you use your Dragon or Tandy in an interesting or unusual way contact John Penn on **04203 5970**

DR71

CLASSIFIED ADS

APOLOGY Owing to the vagaries of the Christmas post we are without our classified advertisements this month Normal service will be resumed as soon as possible Meanwhile an announcement from one of our customers

Hereford (0432) 263476 is offering Dragon software for sale for the price of **£100 ONO** and not £400 as previously stated A Dragon 32 and hardware are also available

HERE'S MY CLASSIFIED AD
(please write your copy in capitals on the lines below)

Name

Address

Tel

Classified rate: 35p per word.

Please cut out and send this form to: Classified Department, Dragon User, 12–13 Little Newport St, London WC2H 7PP.

Write: ADVENTURE

***Pete Gerrard** comes up with some codes for characters*

SO far in this, the only mini-series in the world not to feature Joan Collins (thank heavens), we've managed to introduce fairly simple characters who do little more than be around, and gone on to consider more advanced creations who are capable of contributing significantly to the enjoyment of the game. They are also in most cases rather important in the solving of it was well.

This month we'll be looking at extracts from a real game, and as usual I'll have to make an apology for talking in expanded bits of Basic rather than the real listings themselves. These sadly would make little or no sense if used as stand alone listings, so I hope you'll make allowances for that and I'll try and point out where to make the necessary changes.

We're going to be meeting once again living legend, Dimli Gloing the Wonder Dwarf! Plus a few of his pals, as we're going to consider the introduction of several characters (and the corresponding verbs required) into an adventure game. They're all different, with varying attributes depending on their status in the adventure in question: some are useful, some are just decoration, but we'll sort all that out when we come to it.

By considering a real adventure you should be able to get a better idea of how characters are used and controlled, certainly better than if we just talked in general terms. So, in order of importance we'll be meeting Strombrigner the Grey, Legless the elf, a guide dog, Rolf Harris (how did he get in here?!) and Belin Hey You, of course, take on the role of Dimli Gloing, master of them all, and as such need no introduction. Oh all right then, a brief word. You're a dwarf, and in the game have a specific mission to complete. No treasures to collect, just a series of bizarre solutions. Alas for you it is impossible to complete the adventure without the help of several accomplices, and the first one you'll meet is Strombrigner the Grey.

Strombrigner is the hero of a couple of stories in a magazine that shall be nameless, and all you need to know is that his parents were dyslexic, hence the unusual name. Being a wizard, he is capable of casting spells and generally sorting out one or two things that you can't manage, but he does have this liking for a drop of ale or two. Too little, and he is unable to stop his hands shaking and cannot cast a spell to save his life. Too much, the obvious result occurs and he still cannot cast a spell. So one of your tasks is to ensure that he manages to pay at least one visit to the pub but no more than four. To add to your problems, he's a stubborn old goat, and will sometimes need a bit of persuading in order to comply with any request you might give him.

Figure one shows him making his first and subsequent appearances in the game. As this is an ideas forum rather than a programming course we won't go into too much detail, but you should be able to get my drift, as that strange saying has it.

Two things to note. The subroutine at line 2428 is a simple delay loop to give you time to read a message, and the subroutine at line 5990 extracts and prints a message denoted by the variable me. Right then, let's have a look at Strombrigner.

Line 1082 first of all. This is used if the wizard has never wandered into the adventure before and you are in the correct location for meeting him. CP holds the player's current position, and location 12 being the heart of the pub means that the wizard here. Then we set the wizard wandered flag, ww, the wizard following, wf, and print up descriptions.

Lines 1084 and 1085 introduce a couple more variables, namely ss and vp. vp is used to keep track of the number of visits that he makes to the pub, and ss is used to keep an eye on how long Strombrigner has stayed with you without being given something to do. Now then, if the wizard's following you and the random number falls within a given range and ss is less than 11 then a random message about Strombigner is printed on the screen. Unlike Thorin, the wizard sits down and sings about beer or other things. However if the variable ss is greater than 10 then old Stombrigner gets a fit of the sulks at being given nothing to do and, provided that you're not already in the pub, he wanders off back to it, saying something along the lines of "Well, I'm off to find a decent ale house". Various flags and variables are set as a result of this.

Line 1086 is only used when you go oft to retrieve the wizard, and if you're in location 12, the wizard's already been found once (ww=1), he's not following you at the moment (wf=0) then set the wf flag and print up a message about him reappearing by your side and being ready to join in the game again.

Just four lines of code to give the wizard a real slice of character. He does random things, he sometimes sulks and stomps off to the pub, but when you find him again the shamefaced old boy is ever ready to try and help you out if he can. Of course there's a lot more code involved with Strombrigner than this, because you can talk to him and ask him to do things, but as I've said this is not a programming course. Instead, we're just discussing characters and how they can easily be introduced to your games.

Strombrigner is a meaningful character in that the adventure cannot be finished without him being there to help you, but the next one that we'll look at is just decorative. This is Legolas the elf, always to be found in the pub, and as the game progresses he

Listing one

```
1082 IF CP=12 AND WW=0 THEN WW=1:WF=1:FO
R Q=1 TO 3:ME=Q:GOSUB 5990:PRINT:GOSUB 2
428:NEXT
1084 IF WF=1 AND RND(100)>75 AND SS<11 T
HEN ME=RND(3)+8:GOSUB 5990:GOTO 1086
1085 IF WF=1 AND RND(100)>75 AND SS>10 A
ND (CP<11 OR CP>17) THEN ME=4:VP=VP+1:OB
(17)=0:WF=0:SS=0:GOSUB 5990
1086 IF CP=12 AND WW=1 AND WF=0 THEN WF=
1:ME=5:GOSUB 5990
```

Listing two

```
1080 IF (CP>11 AND CP<18) THEN LL=LL+1:P
RINT:IF LL>10 THEN LL=10:ME=99+LL:GOSUB
5990:GOTO 1082
1081 IF (CP>11 AND CP<18) THEN ME=99+LL:
GOSUB 5990
3452 IF (CP>11 AND CP<18) AND NA=31 THEN
 ME=182+LL:GOSUB 5990
```

Listing three

```
2457 IF NA=61 AND DB=1 THEN PD=0:LO=1:PR
INT"IT DECIDES TO FOLLOW YOU ":OB(61)=-1
:ZZ=ZZ+1:GOTO 10
2458 IF NA=61 THEN ME=204:GOSUB 5990:GOT
O 10
```

rapidly ends up being Legless the elf and the messages used reflect this (**figure two**)

A character should never be used in a game unless he she or it adds something to it You may feel therefore that Legless is a mite redundant but when the game was being play tested one of the testees said that she kept looking around the pub in order to see what Legless was getting up to next completely forgetting about getting on with the adventure in order to keep track of our friendly elf So he fulfils his purpose by adding enjoyment to the game So much so that someone else (hello Sandra!) told me that she wanted to talk to him as well and in a very simple way we can take care of that possibility also

Line 1080 checks first of all to see that you're in the pub and if so the legolas legless variable is incremented by one Since the chap can only handle so much beer we then see how many he's had and print up a suitable message informing the player of the elf's current state of health Once he's had enough he becomes totally incapable of doing anything at all and line 1081 sorts out the final message in the sequence There are ten messages used in total and that seemed to be enough to keep people amused while playing the game

Line 3452 is the talk to legless line which again checks to see whether you're in the pub If you are and you're talking to noun number 31 (legless!) then we use the legolas legless variable ll again in order to extract a suitable message from our file Again you're not really talking to him in the sense that the responses are pre-programmed but it serves to enhance the feeling of genuineness about the elf

The game is capable of being solved without Legless being in there at all but it would be a poorer game without him

We'll draw a discreet veil over the activities of Rolf Harris and Balin Hey (I'll have to start marketing the thing so that you can meet them yourself!) and finish off by looking at the guide dog At first he started off by being like the elf just there for decoration but at a suggestion from someone he grew to play a much more active part in the game My friend said reasonably enough that people are fed up with looking for torches to light their path in a cave so why not use the guide dog in that respect to guide you through the darkness of the caves? Good idea and so that is what this particular character now does

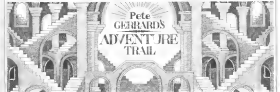
Before using the guide dog you have get hold of it of course and that is the purpose of **figure three** As with Strombrigner there is plenty more code concerning the dog but this should give you some idea anyway The noun variable na is equal to 61 if the player is trying to get the dog If the dog given bone variable db and the played pool variable pp are both equal to one then you can safely take the dog Thus we turn off the pitch dark variable pd turn on the light on variable lo (both of which are checked when the player is deep underground in the inky blackness) and put the dog in the player's possession after printing up a suitable message If either of the two variables pp and db are nto set (and if you want to know why it's necessary to play pool before getting the dog then you'll just have to play the game!) then line 2458 prints up a suitable message

Conclusion

We've only looked at nine lines of code governing three very different characters but that should be enough to give you an idea of what they're about how they work and how they affect the game Also about how various verbs are needed talk for example No character should be put in it if doesn't in some way make for a better game and in their own individual ways Strombrigner Legless and the guide dog all contribute to the enjoyment of the adventure as a whole

Well I hope I've managed to give you an insight into a) how I write adventures and b) what as a player reviewer I'm looking for in other people's games Characters are all important don't neglect them

Bye for now



Pete GERRARD'S ADVENTURE TRAIL

PICTURE the scene if you will It is a frosty morning in Wigan two weeks before Christmas and Our Beloved Editor is already demanding the copy that you're uow reading! There's a black and white rabbit in next door's garden there's an enormous alsatian bounding around two doors away from the rabbit but the animals haven't met each other yet This is not an aid to concentration and I shall continue to report on the situation as the column progresses

Some of you may recall my accqquaintance Professor Deadrock from an earlier *Dragon User* July issue for the curious amongst you Well it looks like he's been out aud about exploring again

Dear Diary

Found myself taking the unusual step of arranging a winter break courtesy of the *Trekboer* Adventure Club What a strange place to begin though people leaving things lying about all over the place Found a manual decided to open it to read it and see if I could make any sense of my surroundings and a key popped out Most odd Naturally enough I then tried unlocking a nearby cabinet No one else around so no-one to see my somwhat furtive actions Have no wish to bring shame on the family name Looked in the cabinet and took everything out of it A strange assortment of items Suddenly (was it something I had done?) I received a strange message and on further exploration found a likely looking robot and read my message Something about the co-ordinates which I suppose I had better take a note for future use

In a burst of inspiration I set the co-ordinates to the number that I had been given 8350 and pressed a distinctly dangerous looking red button Explored still further and found myself in the teleport room Naturally my curiosity got the better of me and I pressed the button in there too and wearing my best suit and carrying a beaker for any strange bits and pieces that might be waiting for me I entered the window that appeared before me I was on a planet at last

By going south and east I stumbled across an unusual ship lying in the middle of nowhere By the simple expedient of going to the hatch I found myself inside it with many objects to take and examine I managed to get a map a shovel a canteen (so much for the beaker) and a cartridge from a helpful robot Thought that was enough to be going along with and after going north and filling my beaker with acid (might as well use it for something) I teleported myself back to the warmth and security of my own ship What a peculiar day's exploration

Had the most fitful of night's sleep, but awoke feeling vaguely refreshed for all that and ready to tackle the next stage of my adventure, whatever that might entail. The cartridge looked as if it were designed to be inserted into the device in front of me, and sure enough so it was. I pressed the button and looked in the drawer until it was empty.

A ladder, eh? I went up it, but by the merest chance I stumbled at the top and managed to spill the acid all over a panel. It dissolved, of course, and I looked around cautiously to see if anyone would notice this accident and come and reprimand me, but these must be dashed exclusive holidays for nobody else appears to be around at all. Was beginning to feel lonely at the thought of it, but I realised that I must continue and as pressing buttons had not done me any harm so far, I pressed the white one that lay before me. So many buttons, so little dangerous effects to date.

Thought I'd better go back and replace the acid that I had been carrying, you never know with stuff like this. If one person uses it then they'll all want it sooner or later, so made a quick trip back to the desert planet and filled up again. Didn't like to drop the beaker anywhere, it looked extremely fragile, but I suppose a cushion or a pillow would break its fall if needs must. I decided to persevere.

Read my map which told me that the co-ordinates for the intriguingly named ice planet were 3816. Entered them into my favourite device and pressed the friendly red button as before. Wearing my trusty suit, which I was now beginning to feel almost at home in, although it's in truth a far cry from the smoking jacket that I normally favour, and carrying a shovel and a blanket I made my way to the teleport room and blundered down onto the surface of the planet.

Holiday on the sun?

One cannot help wondering, dear diary, where all these unusual planets come from, and how these holiday agencies find them and manage to retain them exclusively for us holidaymakers. A desert planet, an ice planet, whatever next one wonders? A garden planet perhaps? We shall see as I find more co-ordinates no doubt. Meanwhile I must carry on with my explorations of this ice planet and see what we can find. Ice, I shouldn't wonder.

After much digging I managed to obtain what appeared to be a sufficient quantity of ice for whatever possible needs I might have for it, and the blanket seemed to be the easiest of things to carry it all in without it slipping away or melting into an embarrassing puddle that would be awkward to explain away at best. On roaming around I found an interesting cenotaph, and although not the athlete that I once was I stumbled and fumbled my way to the top of it. Lo and behold I found an amulet, and couldn't help but think what a splendid souvenir that would make for Mrs Armstrong at the Bridge Club. But I digress.

From the cenotaph I deduced yet another set of co-ordinates. What a lot of numbers to remember for sure, and I am fortunate that despite my advancing years I still have a retentive memory. But time was getting on, and although it might have been merely my dour surroundings I was in dire need of some sleep to awake invigorated for the challenges of what will hopefully be my last journey to another planet. In some haste I made it back to the comfort of my own ship and once more settle down for the night.

Had the strangest dream, all about spiders, but cannot make head or tail of it for there are no such beasts on the ship. I have examined it thoroughly.

Garden centre

As carrying all this was beginning to get awkward I decided to wear the amulet and felt at once strangely reassured by its presence. Having set the co-ordinates in my usual way and pressed the red button I decided that the blanket was a burden that I could well do without, and I dropped it and its ice into a convenient barrel. So much for that! Carrying my beaker carefully this time, I also took along with me a capsule and a rope, wearing as ever my trusty suit, and with the amulet firmly in place I teleported myself onto the surface of this brave new world, and what did I find, dear diary? A garden planet, just as predicted.

On arrival I went east from the bridge and tied the rope to the tree. I could now cross in safety, and almost immediately found myself blundering into a forcefield. Something seemed to be protecting me however, and I glanced reassuringly at the amulet, convinced that it had done the trick. Might as well keep it for myself rather than giving it to Mrs Armstrong. She can have the space suit when I've done with it.

Good lord, a spider! My dreams are all coming true, and in desperation I tried feeding the beast with the capsule. To my relief it appeared to welcome such a feast, although heaven only knows I'm going to send a stiff reprimand to the holiday company on my return for the dreadful food that I personally have had to suffer. Fortunately it would appear that I have not suffered as much as the spider, who ate the capsule and promptly became unconscious. I will admit to feeling a pang of sorrow, one doesn't like to see even the lowliest of creatures in pain, so in my fit of remorse I lurched over to a strange room and left it there. This was after carefully removing what I discerned to be a Xendos plant, which will surely win many a prize at the next horticultural show in the village. That will put a few noses out of joint, I can assure you.

Outside this strange room I pressed a button, and I think that I've somehow managed to kill the spider. This is indeed unfortunate, but one cannot stop to ponder the fate of every animal and insect in the world, and I must hurry on. I stumbled across a grate, but unlike every other grate I've ever met on an adventure holiday this one didn't take too kindly to being opened with a key, or so I deduced. In a fit of pique I jumped up and down, and by the merest of accidents managed to spill some acid on it. I leapt aside hastily and watched in amazement as the grate dissolved just like the panel earlier. However thirsty I get I made a vow never to even think of drinking the acid. Or watering the plant with it, hardy annual or no hardy annual.

Rope trick

On going through the remains of the grate I found myself to my horror in front of a plain of lava. Fortunately there were some pieces of rubble scattered about and I could walk over with ease. Then I had to give myself a pat on the back for my foresight. The rope had nothing to do with the bridge, but was now available for me to climb and make my journey back to my ship easier and safer. What a pleasant surprise, and one up for Deadrock, eh diary?

Once back in my ship I planted the Xendos flower before setting the co-ordinates to 1042 and the journey home to earth. Taking the Xendos flower and going over to and through the teleport window saw the end of my adventure and winter weekend break. Perhaps I'll give the flower to Mrs Armstrong after all. She probably likes flowers.

And this is me again! What unusual adventures the old chap has, eh? Well, no time for dawdling, must get this in the post to Our Beloved Editor. Time only to tell you that the rabbit has now settled down under a large bush, presumably eating its way out from the inside, and the alsatian has vanished into the distance somewhere. Rabbit lives again to fight another day. Bye for now.

Adventure Contact

To help puzzled adventurers further we are instituting an Adventure Helpline — simply fill in the coupon below stating the name of the adventure, your problem and your name and address, and send it to Dragon User Adventure Helpline, 12/13 Little Newport Street, London WC2H 7PP. As soon as enough entries have arrived, we will start printing them in the magazine.

Don't worry — you'll still have Adventure Trail to write to as well!

Adventure ....

Problem ....

....

Name ....

Address ....

....

....

Down in the dumps

This month's screen dump is for a Brother HR-5 printer

THIS program will copy the contents of the screen to a Brother HR-5 printer It can be incorporated into your own programs as a subroutine in the manner shown or used as a stand alone utility to dump screens from other programs

To use the program in the second way DELete lines 10 to 50, change line 190 to END DELete lines 200 onwards RENUMber and save the new version to tape Load and run a program until you see a screen you wish to copy and press BREAK Load the screen dump and RUN The screen will now be printed out

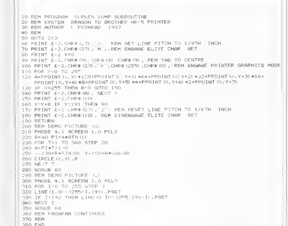
Being in Basic the execution time is rather slow so have a cup of coffee while you re waiting

Note on lines 130 to 150
In theory these should read as follows

130 PRINT £-2 CHR$(A),
140 NEXT X
150 PRINT £-2 CHR$(10)

However if you substitute these lines for the equivelent ones in the program listing you will find that when dumping a pattern or picture that fills the entire screen as in the second example given unwanted characters may be printed at the end of some lines I don t know the reason for this and the only way I ve found to avoid it happening is by sacrificing the final column of pixels Advice please anyone?

K Redhead



```
10 REM PROGRAM  SCREEN DUMP SUBROUTINE
20 REM SYSTEM  DRAGON TO BROTHER HR-5 PRINTER
30 REM AUTHOR  K PEDHEAD  1987
40 REM
50 GOTO 210
60 PRINT £-2,CHR$(27)  1   REM SET LINE PITCH TO 1/9TH  INCH
70 PRINT £-2,CHR$(27), M ,.REM ENGAGE ELITE CHAR  SET
80 PRINT £-2 Y=0
90 PRINT £-2,CHR$(9) CHR$(9) CHR$(9), PEM TAB TO CENTRE
100 PRINT £-2,CHR$(27),"K",CHR$(255),CHR$(0), REM ENGAGE PRINTER GRAPHICS MODE
110 FOR X=0 TO 25"
120 A=PPOINT(X,Y)*128+PPOINT(X Y+1)*64+PPOINT(X Y+2)*32+PPOINT(X,Y+3)*16+
    PPOINT(X,Y+4)*8+PPOINT(X,Y+5)*4+PPOINT(X,Y+6)*2+PPOINT(X,Y+7)
130 IF A=255 THEN A=0 GOTO 150
140 PRINT £-2,CHR$(A), NEXT X
150 PRINT £-2,CHP$(10)
160 Y=Y+8 IF Y<191 THEN 90
170 PRINT £-2 CHR$(27),"2" REM RESET LINE PITCH TO 1/6TH  INCH
180 PRINT £-2,CHR$(18), REM DISENGAGE ELITE CHAR  SET
190 RETURN
200 REM DEMO PICTURE (1)
210 PMODE 4,1 SCREEN 1,0 PCLS
220 R=40 PI=4*ATN(1)
230 FOR T=1 TO 360 STEP 20
240 A=PI*T/180
250 X=130+R*SIN(A) Y=100+R*COS(A)
260 CIRCLE(X,Y),R
270 NEXT T
280 GOSUB 60
290 REM DEMO PICTURE (2)
300 PMODE 4,1 SCREEN 1,0 PCLS
310 FOR I=0 TO 255 STEP 3
320 LINE(I,0)-(255-I,191),PSET
330 IF I<192 THEN LINE(0 I)-(255 191-I),PSET
340 NEXT I
350 GOSUB 60
360 REM PROGRAM CONTINUES
370 REM
380 END
```

Faster, faster, faster!

Gordon Lee wants a quicker run-time. You tell him

Prize

DRAWING conclusions from a Gordon Lee puzzle is a well known way of giving a programmer a pain but drawing anything else can be nearly painless with a copy of David Makin's hot off the duplicator graphics program *Picture Maker*. John Penn Software, David's publisher, are offering 10 *Picture Makers* and ten discount vouchers for this month's winners.

Rules

When you have reduced the Lee Listing to Lightning duration, print out your program, add any notes (bribes will be consumed if not considered), don't forget the tiebreaker, put it all in an envelope marked FEBRUARY COMPETITION and send it to the usual address (on the front cover if any of you are still searching.)

This month's tie breaker is something close to the hearts of many of us: send in any suggestion or suggestions for improving the running times of all or any British Rail train. Quality rather than quantity is the key. Extra points for anything we can't think of an easy British Rail type excuse for.

November winners

THERE's nothing like a bit of literature to empty a dance floor, is there? All we ask you to do is make up a few six letter words, think of a definition and get them accepted at the *Oxford English Dictionary* by last Thursday and what happens? Everybody chickens out except a small and select band whose winning representatives are remembered here in their glory. They are Mark Towlson of Long Eaton, Phil Sapiro of Woolton, Terry Fawcett of Hendon, Paul Weedon of Wotton-under-edge, Andrew Marshall of Bletchley, Graham Barber of Sutton Coldfield, Keith David of Crawley (Three Bridges actually, which sounds a lot nicer) and S A Siddiqui of Chiswick. Invisible medals right away and copies of *Space Trek/Reversi* donated by **R & AJ Preston** eventually.

All the winners came up with at least one answer says Gordon. "Top of the list with 28 words (all in *Chambers*") is Mark Towlson for wheedling out such gems as nilgai, muscle and redeye, though bogeye would probably be a more appropriate term after such a splendid attempt! He also included a list of tiebreakers adding up to 263, including PUBDOG, a pooch on the hooch, ETCHEW, and upper class sneeze, and LINECA, an automatic goal-scoring machine.

Solution

This month's solution should be opposite.

LAST month on this page we were considering an improved method of performing a long multiplication. Such a method is given in **listing 1**, but before examining it in detail there are one or two points to be mentioned. In this program there are only three main string variables: A$ and B$ — the two strings holding the numbers to be multiplied — and Z$, the final product. The basis of this method depends on adding into string Z$ each digit as it is computed, thus eliminating the need to have an array of sub-products.

In order to do this the first requirement is to define Z$ at the outset as a string of zeros long enough to contain the final product.

In addition to these the TIMER is set to zero at the very start of the program to give an indication of the time (in seconds) that any computation needs to be completed.

The method used by the program listing is fairly straightforward. I followed against this outline of the method used in each of the lines.

Lines 20 to 40 — Define initial variables
Lines 50 to 60 — Take each digit in turn (starting from the right) from B$ and convert to variable B
Lines 70 to 80 — Take each digit in turn (starting from the right) from A$ and convert to variable A, calculate position of current value of Z in Z$
Lines 90 and 100 — Find product of A times B (plus any carry). Reset carry and check if V is greater than 9
Lines 110 and 120 — Extract relevant digit from Z$ (Z) and add value V, check to see if new value Z is greater than 9
Lines 130 and 140 — Convert digit Z to string Y$ and insert this character back in its correct place in string Z$

Listing 1

```
 10 TIMER=0
 20 A$="7129076934"
 30 B$="2763128729"
 40 C=0:L=LEN(A$+B$):Z$=STRING$(L,"0"):A$="0"+A$
 50 FOR M=LEN(B$)TO 1 STEP-1:D=LEN(B$)-M
 60 B=VAL(MID$(B$,M,1))
 70 FOR N=LEN(A$)TO 1 STEP-1:E=LEN(A$)-N
 80 A=VAL(MID$(A$,N,1)):P=D+E
 90 V=A*B+C:C=0
100 IF V>9 THEN C=INT(V/10):V=V-C*10
110 Z=VAL(MID$(Z$,L-P,1))+V
120 IF Z>9 THEN Z=Z-10:C=C+1
130 Y$=STR$(Z):Y$=MID$(Y$,2)
140 Z$=LEFT$(Z$,L-(P+1))+Y$+MID$(Z$,(L+1)-P)
150 NEXT N
160 NEXT M
170 IF LEFT$(Z$,1)="0"THEN Z$=MID$(Z$,2)
180 PRINT Z$
190 PRINT"Time taken ";TIMER/50;"seconds"
```

Now any two numbers having *n* and *m* digits respectively will, when multiplied together, result in a product having either *n* plus *m* digits or one less than this number. Consequently, having defined variables A$ and B$ in lines 20 and 30, the total length of Z$ can be defined in line 40. In the listing, as both numbers have ten digits each, the value of L will be given as 20 and the STRING$ command which follows (line 40) will produce an initial string for Z$ as twenty zeros. The addition of the extra zero to the end of A$ in this line simply to facilitate the handling of any final carry at the end of the calculation.

To clarify the working of the program the variables used are listed below.

A$, B$ The two numbers to be multiplied
Z$ The final product
C The carry variable
L Length of A$ and B$ combined, ie length of Z$
M Position of digit in B$ being operated on
N Position of digit in A$ being operated on
A Value of a digit from A$
B Value of a digit from B$
Z Value of a digit from Z$
D Relative magnitude of current value of B (ie 0=units, 1=tens)
E Relative magnitude of current value of A
V Product of A and B plus any carry
P Relative magnitude of current value of Z

If the string Z$ contains a leading zero at the left hand end this is removed at line 170 before the result is printed out together with the time taken to complete the computation. On the Dragon the TIMER variable counts fifty times per second so if it is reset to zero at the start of the program the expression TIMER/50 will give the total running time in seconds.

We are now in a position to use this program to try out a much larger computation. In **listing 2** values at lines 20 and 30 (from **listing 1**) have been redefined to two 50 digit numbers to be multiplied together. Because of the extra string space needed it will also be necessary to include a CLEAR 500 command in order to reserve more bytes for string storage. This can be added to line 10 as follows:

10 TIMER=0:CLEAR 500

Listing 2

```
10 TIMER=0
20 A$="7379530434425272663339014735897481811475521184535 1"
30 B$="3065854077584954188367235930222177586032449936553 9"
   [22624569376669338122753248773577267597286450767948329
    20273071442749519791057202895364969936886759189]
```

If the program is now run with these new values the running time is much longer than before a time in the region of 278 [illegible] answer is that given in square brackets below **listing 2** so you can check your answer This month s competition is to devise a program to perform the multiplication from **listing 2** but in such a way as to improve on the program's running time as given above

This can be either a modification or a listing of your own devising It should be written in Basic and the timer set to zero as [illegible] seconds should be the last instruction executed by the program Note that the use of any speed poke is NOT allowed

On a pratical level the longer the sequence of digits to be operated on the faster we would wish the computation to be At the start of this article last month I said that it is possible to perform computations with thousands of digits Of course since there is a maximum [illegible] of 2[illegible] characters in an one string variable, it would be necessary to modify any listing to break longer numbers into convenient blocks of say 250 characters

This technique offers many interesting opportunities to perform otherwise lengthy computations by computer

The Answer

This is Gordon Lee s own solution to the January competition see page 26 for results

ANSWER This words on the following list all sum to 263 when operated on in the manner described

Aching cheesy dragon homage itches mickle mobile rename sacred sailed scalps seared socked spices steams Sweden

SOLUTION As there are over 308 *million* permutations of six different letters it is perhaps suprising that only a handful of these total the value 263. It is unlikely that even allowing more obscure words than those on the list above the total number will exceed two dozen Needless to say it is extremely imprectical to examine all of these 308 million possibilities to find the acceptable words The program as listed uses a number of short cuts

Firstly the words are built on the list of two letter starters as held in the data lines These are all of the two-letter combinations with which a word can begin For example if a word begins with the letter M the second letter can only be an A E I O U or Y By using this technique a lot of dead ground can be eliminated

The actual value that is produced by any word can be calculated in a single operation If the values of the six letters in the word are represented by the algebraic terms a b c d e and f the final total will equal the expression a + 5b + 10c + 5e + f To understand this consider the second row of numbers There are five terms which equal from left to right (a+b) + (b+c) + (c+d) + (d+e) + (e+f) If this process is continued for the other four lines the final expression can be obtained

Once the first two letters have been read from the data statements the remaining four letters are generated in sequence using the three loops C D and E and the variable F (line 120) The total letter values cannot exceed 263 so once the first two letters have been determined their final contribution to the total (a+5b) is subtracted from 263 (line 60) This will indicate the upper limit for the third letter in the word (c) and the appropriate loop can be set at line 70 Again once the first three letters have been selected a similar procedure can be applied for the fourth and fifth letters (lines 80 to 110) The final letter f is found by taking the residue of 263 minus the total value produced by the other five letters (line 120) If this is not in the range 1 to 26 (ie A to Z) the program jumps to line 160 In this way only letters which do not cause the total to exceed 263 are considered Remember that the central two letter values have to be multiplied by 10 so the further that they are into the alphabet the less likely they are to prove valid

All possible letter combinations produced by the above arrangement are printed eight at a time on the screen Pressing the space bar will display the following eight There is one further refinement to the program which allows certain permutations to be jumped without being displayed This operation has an effect on the *third* letter of the word If instead of pressing the space bar one of the letter keys is pressed the display will jump immediately to words having the letter indicated at third position and the sequence will continue from there For example suppose that the program has listed words beginning with CR and has reached CRB Clearly there are no words beginning with CRB, CRC or CRD so pressing E takes the permutation to sequences beginning CRE

By using this technique all permissible letter combinations can be scrutinised and acceptable words extracted from them

Listing A

```
[illegible]
[illegible]
[illegible]
[illegible]
  5 [illegible] LEFT$ [illegible] RIGHT$ [illegible]
[illegible]
  7 [illegible]
  8 [illegible]
  9 [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
140 S$=(CHR$(A+64)+CHR$(B+64)+CHR$(C+64)+CHR$(D+[illegible]+CHR$(E+64)+CHR$(F+64)
150 PRINT S$ [illegible] PRINT T [illegible] IF T=9 THEN GOSUB 20[illegible]
160 IF F$ [illegible] THEN [illegible] ASC(F$ [illegible]
170 NEXT E,D
180 GOTO 40
[illegible]
[illegible] Z$=INKEY$ [illegible]
[illegible] IF Z$<> [illegible] THEN F$=Z$
[illegible] CLS:PRINT:RETURN
250 CLS:PRINT:RETURN
[illegible] DATA AA,AB,AC,AD,AF,[illegible]
BA,BE,BI,BL,BO,BR
1010 DATA BU,BY,LA,CL,CH,[illegible]
[illegible]
[illegible] DATA [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
```

TURBOCHARGE YOUR DRAGON:

With our great value hardware and software:

BASIC 42

Extended BASIC for the Dragon 64

For Dragondos (please state version) £19.95

Run your Dragon in 64K mode while retaining BASIC and DOS. Print on hi-res screen using standard PRINT commands, and a screen layout of 24 rows of 42 columns. Other features include

Alternative, redefinable character sets, control key for special characters, repeating keys and commands in lower case, windows, CATCH command for automatic return to window, inverted video (green on black/black on green), true underlining and extra PRINT commands and functions

LIBRARY sets commands and functions. Automatic startup of BASIC program. TEXT command for software compatibility. Still 23335 bytes free to BASIC. Patches for Dragondos 1.0 Contained in extra UTILITIES from disk

HELP UTILITY £5.00
Extensions to BASIC 42 include change cursor characte, scroll disable, pause listing, BREAK disable, improved TRON (allows single stepping). Detailed help and error messages.

SPOOL UTILITY £5.00
Use computer while printing. 35K print buffer TYPIST program turns Dragon into typewriter

ICONS UTILITY £5.00
Put icons in your program! Controlled by cursor or mouse! Commands to define, clear, load and save icons, pens and windows

STRUCTUR UTILITY £5.00
Another first. Structured BASIC on the Dragon! Allows named procedures, improved loop controls by WHILE WEND, and REPEAT UNTIL etc

DOS UTILITY £5.95
Make friends with your DOS. Enter all the main DOS commands, plus LIST EDIT etc. and select files by cursor or mouse

* * * **NEW** * * *

KLIK UTILITY £19.95
Point and click operation of entire system by keyboard or mouse with pull-down menus, pointer, dialogue box, control buttons, and help messages

Selective directories, files as icons, repeating DOS commands, improved word processor/text editor with trace and pause facilities. Set-up module for easy control of screen windows, BREAK key etc. Desktop accessories: disk based spool, memo pad, snapshot, jotter etc. Klik BASIC: write your own windows, icons, pull-down menu programs

HARDWARE

Superdos Cartridge	£75.00
40/80 Track Drives inc. Cartridge	
Single Drive (180 – 720K)	£189.95
Dual Drive (360 – 1440K)	£289.95
Superdos controller (chip only)	£10.00

DISK SOFTWARE FOR DRAGON 32/64/128
WITH DRAGONDOS/CUMANA DOS 2.0

NEW

Pixie (Mindsoft) £14.95
Icon-driven drawing program. Requires joystick

DSKDREAM (Grosvenor) £19.95
The standard Dragon Editor/Assembler

D.R.B. (Grosvenor) £9.95
Machine code database program

SOURCE MAKER (Pamcomms) £9.99
Disassembler for use with DSKDREAM

DISK-KIT (Pamcomms) £9.95
Sort out your disk problems

MACGOWAN SOFTWARE

PRINTER CONTROL A text AND graphics processor	FROM	£19.95
DUMPER* Relocatable screen dump program	FROM	£5.45
COLOR PRINT PMode 3 screen dump program	FROM	£6.60
NEW		
STARLITE LIGHTPEN S/W Upgrade includes screen dump	CASS DISK	£7.00 £9.00
MONITOR/ASSEMBLER* Printer orientated	CASS DISK	£12.00 £15.00

*Prices vary according to printer, please specify

MONEYBOX (Harris) £14.99
Home and small business accounts

MAILBOX (Harris) £16.99
Selective mailing list program

SHAREBOX (Harris) £19.99
Manage your stocks and shares

SALESBOX (Harris) £19.99
Balance B/F Sales Ledger

BILLSBOX (Harris) £19.99
Balance B/F Purchase Ledger

CASHBOX (Harris) £19.99
Double-entry Nominal Ledger

STOCKBOX (Harris) £19.99
Full-featured Stock Control

ORDERBOX (Harris) £15.99
Invoicing linked to Sales or Stock

Cheques/PO s/Further details/dealer enquiries to

HARRIS MICRO SOFTWARE

49 Alexandra Road, Hounslow Middlesex, TW3 4HP   Tel (01) 570 8335

JOHN PENN DISCOUNT SOFTWARE

SALE BARGAINS (limited numbers only)

Radio Shack cartridges for Dragon & Tandy

COLOR FILE — a useful filer/diary with 21 page A4 manual	WAS £10.00	NOW £5.00
COLOR CUBES many variations on the Rubic Cube theme with 32 page A4 Manual	WAS £10.00	NOW £4.00
ROMAN CHECKERS — a challenging game	WAS £5.00	NOW £3.00
ART GALLERY — draw colour images on screen — with 22 page manual and cassette. TANDY ONLY	WAS £10.00	NOW £4.00
GIN CHAMPION WITH 19 page manual	WAS £10.00	NOW £4.00
WILDCATTING — an oil speculating game	WAS £8.00	NOW £3.00
BACKGAMMON	WAS £10.00	NOW £4.00
BUST OUT	WAS £9.00	NOW £3.00

GAMES

NEW FORMULA ONE (Pamcomms) cass. or disc Split race game	£8.95
TOTAL ECLIPSE (Eclipse/Femmar) cass	£6.50
LARKSPUR WALDORF IS TRAPPED (Prestosoft)	£3.50
FOOTBALL MANAGER (Addictive)	£4.00
NUMEROLOGY (Occult)	£5.00
THE 13TH TASK (Arc) text adv	£3.00
COLOSSAL CAVE (Cowen) text adv	£5.00
HARE RAISER (Haresoft) double pack (gr. adv.)	£5.00
PREDICTOR (Bentley)	£4.00
TIM LOVES CRICKET (Peaksoft)	£4.00
SAS (Peaksoft)	£2.00
PHOTO FINISH (Peaksoft)	£2.00
BACK TRACK (Incentive)	£3.50
MANIC MINER (Software Projects)	£3.50

BARGAIN BASEMENT
£1.50 each, five for £6.00, ten for £10.00
Please give at least two alternatives if possible
DOMINOES, GOLF, CHOCOLATE FACTORY, DON'T PANIC, OSSIE, MINED OUT, NORTH SEA OIL, DEATH'S HEAD HOLE, AMAZING DRAGON TREK, SUPERSPY, SURPRISE, STOCKMARKET, GRID RUNNER, NIGHT FLIGHT, CHAMPIONS, BOPSWIZZLE, DRAG RUNNER, TRANSYLVANIAN TOWER, LEGGITT PETTIGREW'S DIARY

UTILITIES/BUSINESS SOFTWARE

NEW David Makin's PICTURE MAKER. Allows you to design and manipulate PMODE 3 graphics in a variety of ways most of which require selection of a screen position	£5.00
MUSIC MAKER (Makin)	£5.00
DOUBLE PACK OF PICTURE MAKER AND MUSIC MAKER	£9.00
DISK-IT (Pamcomms)	£10.00
SOURCEMAKER (Pamcomms)	£8.00
HI RES — TEXT (Starship)	£3.00
PERSONAL BANKING SYSTEM (Hilton)	£9.95
OS software for Dr 64. OS9 Operating System and disc drive(s)	
BASIC 09	£21.00
'C' COMPILER	£27.00
R.M.S.	£20.00
PASCAL, CASH BOOK & VAT, STOCK RECORDING	£15.00 each
OS9 MANUALS for BASIC 09, CASH BOOK & VAT STOCK RECORDING	£2.50 each

EDUCATIONAL CORNER

Dragon Data series: NUMBER PUZZLER, CIRCUS ADVENTURE	£2.00 each or two for £3.00
Dragon Educational series: NUMBER CHASER	£2.00
Shards/Cambrian/Tiger software: TIGER GRAND PRIX, FUN TO LEARN, FAMILY PROGRAMS, MONSTER MATHS, FUN & GAMES, SNOW QUEEN, QUIZ PACK, INFANT PACK	£1.50 each or three for £3.00
Ampalsoft (Cheshire Cat) series: MATHS LEVEL I, MATHS LEVEL II, 'O' LEVEL MATHS, SUPERSPY, BASIC TUTOR — BEGINNERS LEVEL, ADVANCED LEVEL	£3.50 each or two for £6.00

PERIPHERALS

SAM chips (74LS783)	£15.00 each

HOW TO ORDER

Write to us (or phone with an Access order) giving the titles of the programs you would like. Please include 50p for postage/packaging on single orders and 75p for two or more items ordered. Postage to Europe is £3.00 and £6.00 to rest of world (surface rate). Please could you give a telephone number if possible as well as your name and full address. We try to despatch orders within 24 hours, but allow up to 21 days if necessary.
Cheques/postal orders made payable to **JOHN PENN DISCOUNT SOFTWARE**

JOHH PENN DISCOUNT SOFTWARE, DEAN FARM COTTAGE, KINGSLEY, BORDON, HANTS GU35 9NG
Tel Bordon (04203) 5970
For all your Dragou hardware requirements coutact Harry Whitehouse ou Telephoue 0636 705230

DR72